이학박사 학위논문

Fire Detection and Semantic Fire Image Segmentation using Deep Learning

(딥러닝을 이용한 화재 감지 및 화재 이미지 시맨틱 분할)

2020년 8월

서울대학교 대학원

협동과정 계산과학전공

송 경 민

Fire Detection and Semantic Fire Image Segmentation using Deep Learning

(딥러닝을 이용한 화재 감지 및 화재 이미지 시맨틱 분할)

지도교수 강 명 주

이 논문을 이학박사 학위논문으로 제출함

2020년 4월

서울대학교 대학원

협동과정 계산과학전공

송 경 민

송 경 민의 이학박사 학위논문을 인준함

2020년 6월

위 원 장 _____ (인)

부 위 원 장 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

위 원 _____ (인)

Fire Detection and Semantic Fire Image Segmentation using Deep Learning

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Kyungmin Song

Dissertation Director : Professor Myungjoo Kang

Interdisciplinary Program in Computational Science
and Technology
Seoul National University

August 2020

Abstract

Recently, deep learning has become the most important and powerful topic in various research fields. It has shown excellent performance in image classification and has been applied to the fields of object detection and semantic image segmentation of computer vision. In this thesis, we proposed deep neural networks suitable for fire image detection and segmentation tasks with excellent performance. In addition, we proposed a small-sized network for fire image segmentation based on squeezed deep-learning techniques and applied it to an embedded device. Several extensive experiments are presented to demonstrate its better performance compared with the existing methods for fire detection and image segmentation.

Key words: semantic image segmentation, object detection, deep learning, fire image, squeezed deep learning
Student Number: 2014-31032

Contents

List of Figures

LIST OF FIGURES

List of Tables

Chapter 1

Introduction

Fire accidents can result in fatality apart from injury and property damage. Quick fire detection and warning is the best way to reduce fire damage. Therefore, maintaining an optimum response time to a fire accident is a very important factor in fire response systems.

However, current fire prevention and segmentation systems find it difficult to respond quickly to the early stages of fires. In addition, when a person monitors the system in real time, the demand for manpower is too high and maintenance costs are exorbitant when using existing fire segmentation equipment across large areas.

Existing fire detection methods are categorized according to fire sensors (smoke, flame, temperature, etc.) [2, 23, 6] and image processing [40, 5]. Sensor-based fire detection has the disadvantage of degrading the performance of the system based on various factors of the surrounding environment. For example, fire detection using a smoke sensor may not be effective when air spreads around the sensor, and use of a temperature sensor can delay fire detection if the ambient temperature is already high. In the case of flame sensors, ultraviolet light may be absorbed by smoke or other factors, reducing sensitivity.

However, fire detection based on image processing can solve many of the above-mentioned problems with sensor-based fire detection. In particular, it is possible to minimize additional installation costs by using the existing CCTV cameras already installed, and the dispatch costs incurred by false

alarms can be reduced. However, conventional image-based fire detection methods are difficult to apply to real situations due to empirical and experimental threshold settings, which can generate a false alarm for flame-like objects. In addition, the conventional method is difficult to apply in case of motion between image frames, i.e., video data.

Moreover, in automated devices such as drones, which have recently been adopted for fire monitoring, research on how to find an area of flame accurately within a wide area around the occurrence of a fire is necessary[50]. Therefore, studies have been conducted on segmenting fire zones. Premal *et al.* [39] proposed a technique for extracting an area of flame in 2014 using YC_bC_r transformation and the relationship between luminance and chrominance in RGB images; Tuba *et al.* [54] proposed a more accurate flame region extraction method by improving this algorithm. However, the revised method is also difficult to use because of its poor performance.

Recently, with the development of deep-learning technology, there are many cases in which these techniques are applied to existing image-based application systems. In the field of fire detection, attempts have been made to apply convolutional neural networks (CNN), which perform well in image processing [36, 27, 1]. CNN has the advantage of being able to extract features through training by accepting the original image without complex calculations such as algorithm-based feature extraction or preprocessing.

The CNN method is also frequently applied in image segmentation to classify the label for each pixel of an image. Semantic image segmentation is a widely used technique in many fields, such as general images (Common Objects in Context (COCO) datasets [29]), face analysis [37], road analysis [26], satellite images [58], medical image analysis [45, 41, 9], etc. Image segmentation using deep learning was originally based on fully convolutional networks (FCN) [31], but the performance was inadequate. To increase performance, various models such as U-Net [45] and FusionNet [41] have been proposed. As these deep-learning techniques have become more accurate, the number of weights has increased, requiring a large amount of computation and memory. However, these large models are difficult to apply practically, so research such as SqueezeNet [21] has been conducted to reduce the amount of computation and required memory.

In this thesis, we propose a fire detection and fire area localization process, a binary segmentation that determines the fire area and its compression model using a deep-learning technique. For fire detection and localization, our model was verified based on various datasets classified into fire images and non-fire images. For segmentation, we also verified our proposed method by experimenting with the FiSmo database [4], written to determine the presence of fire in the images and segmentation mask of the particular region, and the Corsican Fire Database [53].

This thesis is organized as follows. In Chapter 2, we introduce the basic concepts of deep learning required for our proposed network. In Chapter 3, we propose a network for classifying images of fire scenes and localizing the fire. In Chapter 4 and 5, we propose a binary segmentation network for fire images and its compressed model. Finally, Chapter 6 presents the conclusion. The experimental environment for this thesis is detailed in **Table 1.1**.

Hardware	**Specification**
CPU	Inter Core i5-6500
GPU & memory	TITAN RTX & 24 GB
Software	**Version**
Python	3.7
TensorFlow	1.15.0
Keras	2.3.1
Ubuntu	18.04

Table 1.1: Experimental environment.

Chapter 2

Preliminaries

In this chapter, we provide an overview of the basic concepts necessary to understand our proposed deep-learning methods. First, we introduce widely used architectures for image classification and explain the function and training method for which they have been used. Subsequently, we introduce semantic segmentation networks that use deep learning, and lastly the compression model of deep learning.

2.1 Image classification

AlexNet [25] is a CNN structure that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) competition in 2012. It reduced the error rate by a greater amount than image classification using machine-learning or image-processing techniques. Since then, deep learning has been recognized as more powerful than any existing methods of image classification, and it has been used consistently.

As shown in **Figure 2.1**, AlexNet consists of eight layers, including five convolutional layers and three fully connected layers. The unique feature of the network is that it is designed in a parallel structure to perform a parallel operation with two GPUs. The second, fourth, and fifth convolutional layers connect only those that are collinear in the feature map of layer, whereas the others connect to all feature maps of the previous layer. It takes the form of $224 \times 224 \times 3$ input images and the first convolutional layer has 96 feature

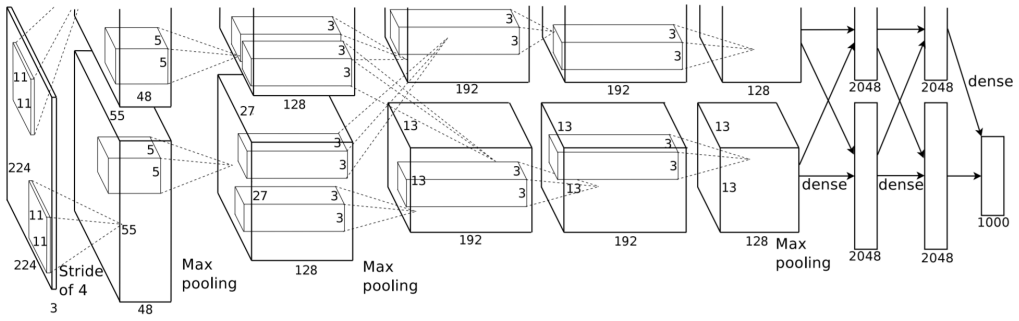

Figure 2.1: AlexNet architecture

maps using an $11 \times 11 \times 3$ kernel. In each step of the convolutional layer, the rectified linear unit(ReLU) function was added as an activation function to ensure nonlinearity, and zero padding was added during convolution to prevent the feature map from shrinking. This operation is performed by reducing the size of the feature map to 7×7 using pooling after the convolution operation at the first, second, and fifth layers.

The pooling operation includes max pooling, average pooling, etc. It can also adjust the size of the feature map using a stride like the convolution operation, as shown in **Figure 2.2**. After this, it goes through two fully connected layers with 4096 neurons and finally obtains its final output via the softmax function.

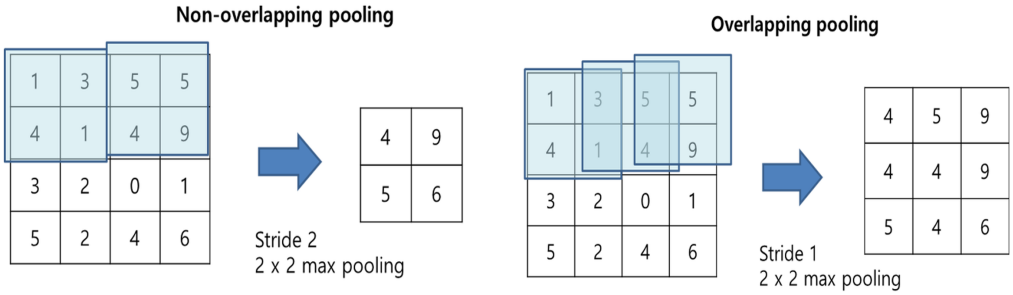

Figure 2.2: Pooling operation example.

As mentioned, regarding the overall structure of AlexNet above, there are 60 million parameters in this network. As there are too many parameters,

an overfitting problem that has a good performance for training data but poor performance in validation data may occur. The easiest method to avoid overfitting in image data is to increase the amount of training data through augmentation, which is a technique to artificially create new training data from the existing data. Data augmentation consists of generating rotation, brightness, zoom, horizontal shift, vertical shift, flip, crop, etc. as shown in **Figure 2.3**.

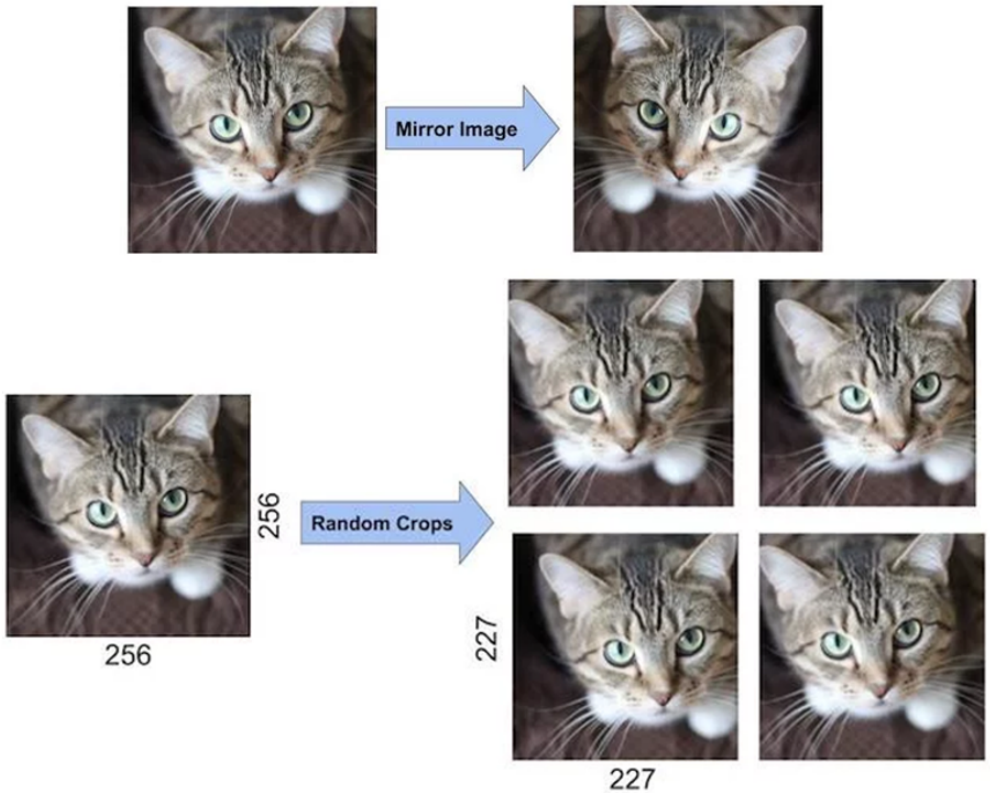

Figure 2.3: Data augmentation example.

Another approach to avoid overfitting is to exclude some neurons with a certain probability from training, using a dropout as shown in **Figure 2.4** (b). The dropout is applied only during training, and in the actual test all neurons are used as shown in **Figure 2.4** (a). Therefore, we do not train entire neurons in the layer that applies dropout and this helps prevent over-

fitting.

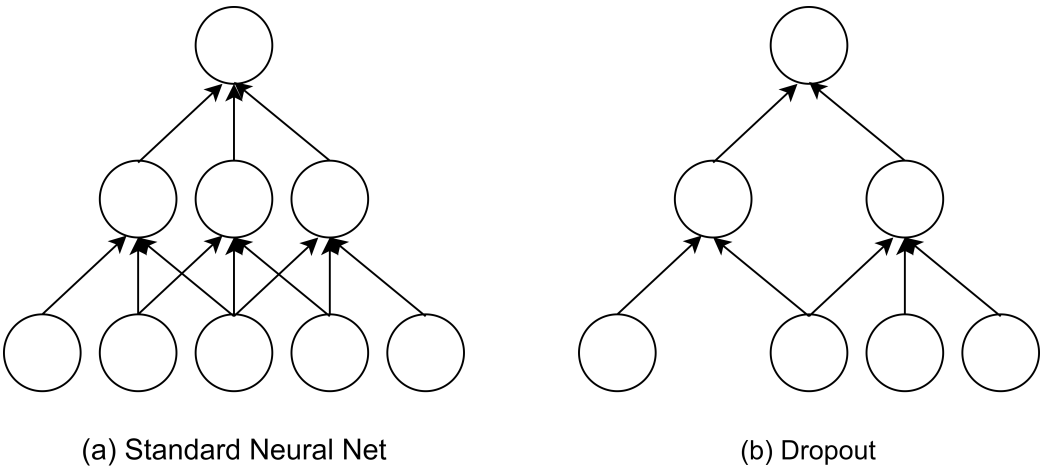

Figure 2.4: Dropout.

Starting with AlexNet in 2012, models based on deep learning were consistently announced at the ILSVRC competition in every subsequent year, and models such as ZFNet [62], VGGNet [49], GoogLeNet [51], ResNet [17], etc. were introduced.

ZFNet is similar to AlexNet, but it improves AlexNet by analyzing it using Deconvnet and modifying the kernel size. It has also enhanced the understanding of deep learning, which was previously perceived as a black box through feature map visualization. In the case of VGGNet, the accuracy is improved by using a deeper and smaller kernel size than the previous network. GoogLeNet incorporates an inception module that can reduce the amount of computation through 1×1 convolution, which prevents overfitting and vanishing gradient problems. As shown in **Figure 2.5**, the inception module was able to reduce several parameters by using 1×1 convolution before 3×3 convolution or 5×5 convolution, which can increase the number of parameters.

ResNet stacked the network much deeper than the existing networks and discovered degradation problems that deteriorated test performance as well as training performance, not overfitting and vanishing gradient problems. To solve this problem, ResNet proposed a residual block as shown in **Figure 2.6**.

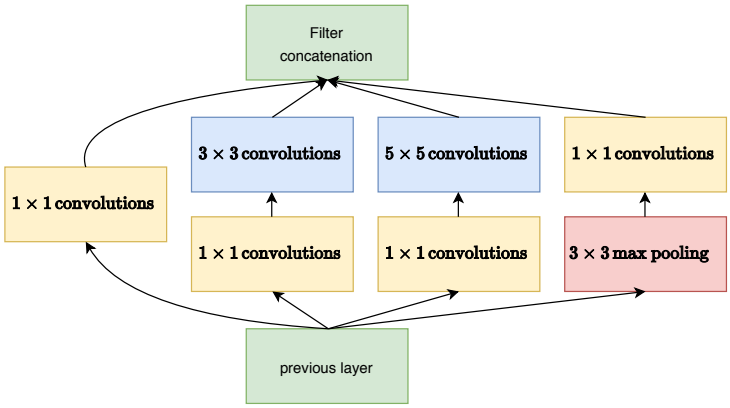

Figure 2.5: Inception module.

By feeding an identical mapping from the previous layer, it is easier to optimize the residual mapping than to optimize the original. In addition, Veit *et al.* [56] demonstrated in 2016 that the residual block has an effect similar to that of building an ensemble model.

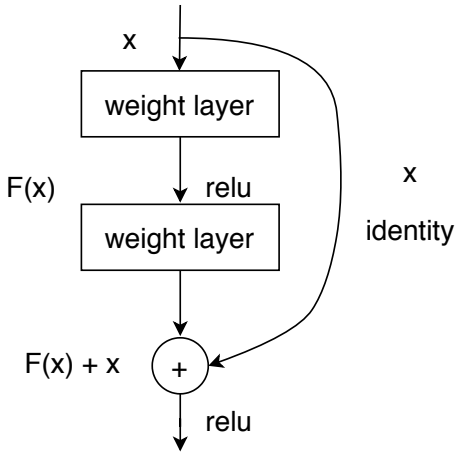

Figure 2.6: Residual block.

The inception module proposed in GoogLeNet and residual block proposed in ResNet have been used so far. Based on the abovementioned modules and blocks, Densenet [20], ResNeXt [59], SENet [19], and EfficientNet [52] have been proposed and the top-5 error rate on ImageNet has been reduced

to 2.3%.

2.2 Object detection

Object detection consists of multi-label classification, which predicts multiple classes in an image, and bounding-box regression, which predicts the bounding-box location information of an object. Object detection using deep learning can be divided into two methods: a two-stage object detection method that sequentially performs two processes of region proposal to localize an object and a corresponding region classification, and a one-stage object detection method that performs the above two processes at once.

Regions with CNN (R-CNN) [15] is a representative method of two-stage object detection, which has been developed into Fast R-CNN [14] and Faster R-CNN [44] with improved performance. Faster R-CNN replaced selective search with a region proposal network (RPN) to resolve inference and bottleneck problems. **Figure 2.7** shows the overview of the Faster R-CNN, which is a two-stage model that carries out RPN first to propose a region and classify the class, and modifies its bounding-box position information.

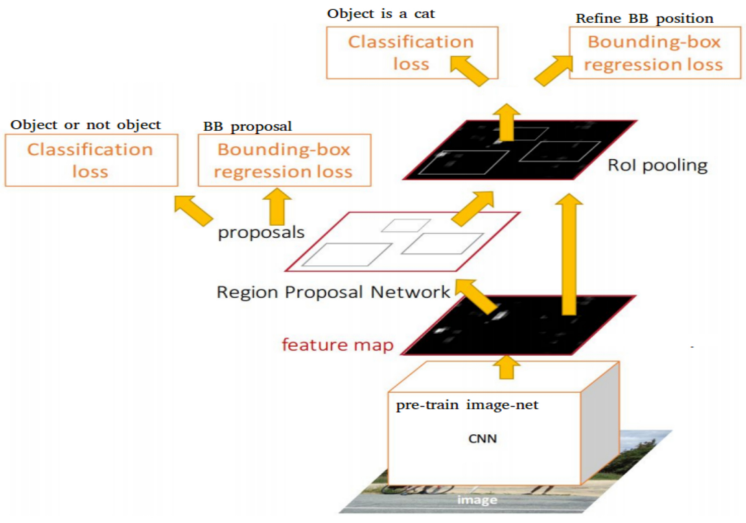

Figure 2.7: Faster R-CNN architecture.

The RPN is implemented using convolution, and the input value uses feature maps extracted from the previous base network. To create region proposals, a 3×3 spatial window is slid over the feature maps created in the base network. For each point of the sliding window, multiple region proposals are predicted at a time. The highest number of region proposals is denoted by k, which is referred to as an anchor. Normally, nine anchors are used for each sliding window point; three different aspect ratios and three different scales are combined, all of which have the same center point (x_a, y_a). The depth of the feature map extracted from the sliding window becomes the lower dimension, and through the classification layer, two predictions are given as probability values for whether the anchor is an object or not. In the regression layer, four values of $(\triangle_x, \triangle_y, \triangle_w, \triangle_h)$ of the anchor are derived and final proposals are obtained. The overall structure of RPN is shown in **Figure 2.8**.

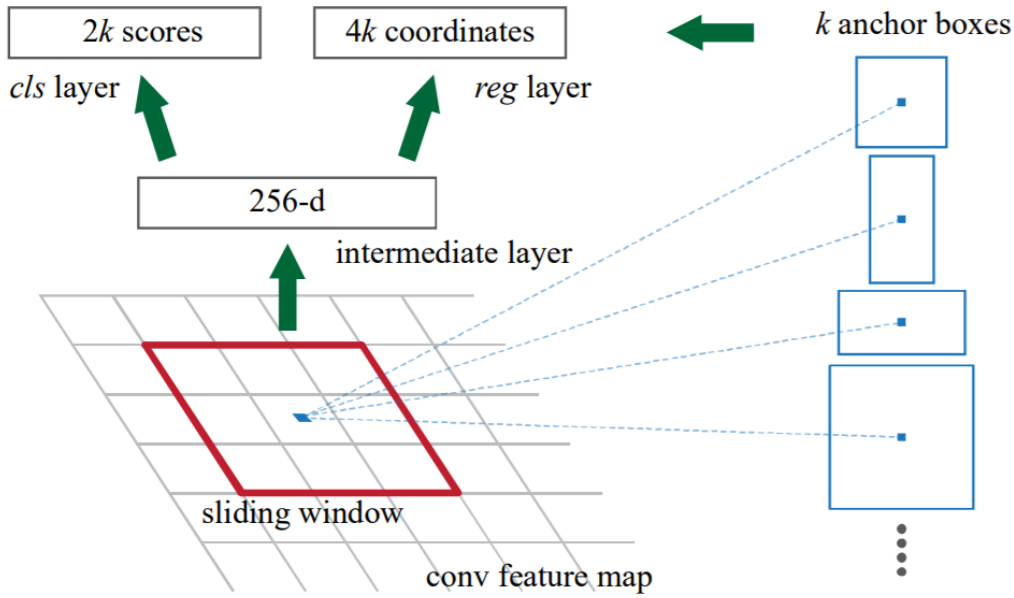

Figure 2.8: Region proposal network (RPN).

The RPN provides the output values for the proposed regions of different sizes. However, to perform classification, the size of the feature map must be the same. To solve this problem, feature maps of different sizes are converted

to the same size using Region of Interest (RoI) pooling. In addition, to improve performance, when multiple boxes overlap, non-maximum Suppression (NMS) is used to select the most certain one, and hard negative mining is used to reduce the negative samples (background).

Single Shot Multibox Detector (SSD) [30] and You Only Look Once (YOLO) [42] are representative methods of one-stage object detection. This has the advantage of being fast because it performs one step lesser than Faster R-CNN, but it is lesser accurate. However, due to its high speed, YOLO has advanced to v3, and SSD has also been proposed for follow-up studies such as Deconvolutional Single Shot Detector (DSSD) [12] and RetinaNet [28]. One of the representative techniques of the one-stage method, SSD, is relatively simple compared to the R-CNN technique. The SSD model added several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios. It associated default bounding boxes with each feature map, for multiple feature maps at the top of the network. The default boxes tile the feature map in a convolutional way, so that the position of each box is fixed. For each feature map, it predicts the offsets relative to the default box shapes, as well as the per-class scores that indicate the presence of a class in each of those boxes. The details of the SSD model are shown in **Figure 2.9**.

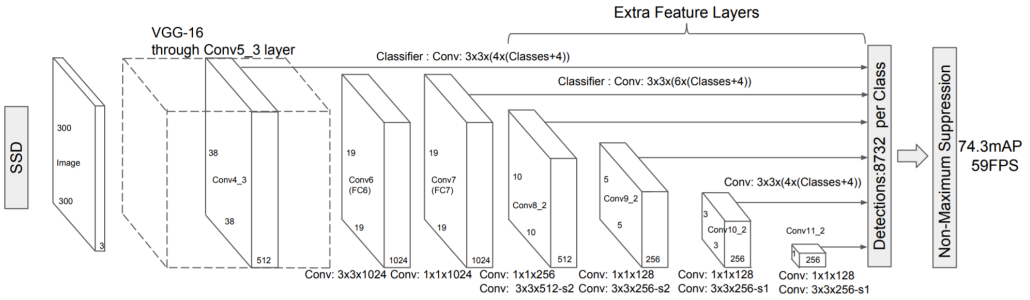

Figure 2.9: Architecture of Single Shot Multibox Detector.

2.3 Semantic image segmentation

FCN [31] is the first model to be applied to a semantic segmentation task using a deep-learning model that shows excellent performance in images classification, as shown in **Figure 2.10**.

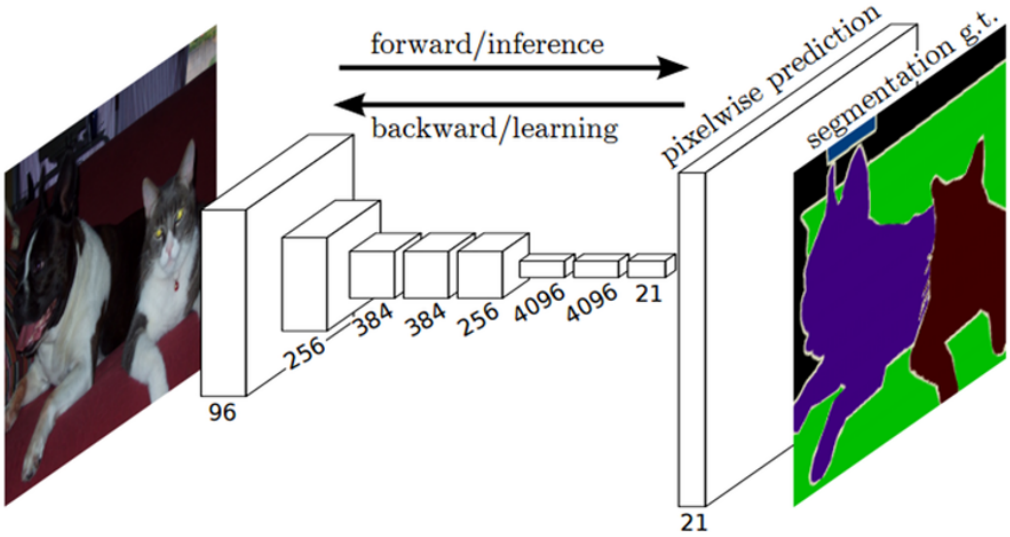

Figure 2.10: Fully convolutional networks.

That is, it uses a good network (AlexNet, VGGNet, GoogLeNet, etc.) that has been proven in the image classification mentioned above. The noteworthy aspect of FCN is that the fully connected layer can be considered as a 1×1 convolution. If we consider this view, we can maintain location information and call it convolutionalized. Its advantage is that because all operations in networks are convolutional, they are no longer limited by the size of the input image. In addition, it is possible to process the entire image at once instead of in units of patches, which can reduce the computation of overlapping parts.

As the purpose of semantic segmentation is to provide dense prediction for all pixels in the image, it is necessary to restore the sizes of these coarse feature maps to the original image size. The process of restoring to this original image size is called upsampling. Upsampling increases the coarse feature

maps corresponding to each class to their original size. The upsampled feature maps are then combined to form the final segmentation map.

However, segmentation maps obtained by simple upsampling to the original size from a coarse feature map of 21×21 often miss details of information. FCN improved the performance by using the concept of the skip layer, as shown **Figure 2.11**. This adds upsampling prediction values from the third and fourth pooling feature maps, which are less coarse, to the $32\times$ upsampled prediction.

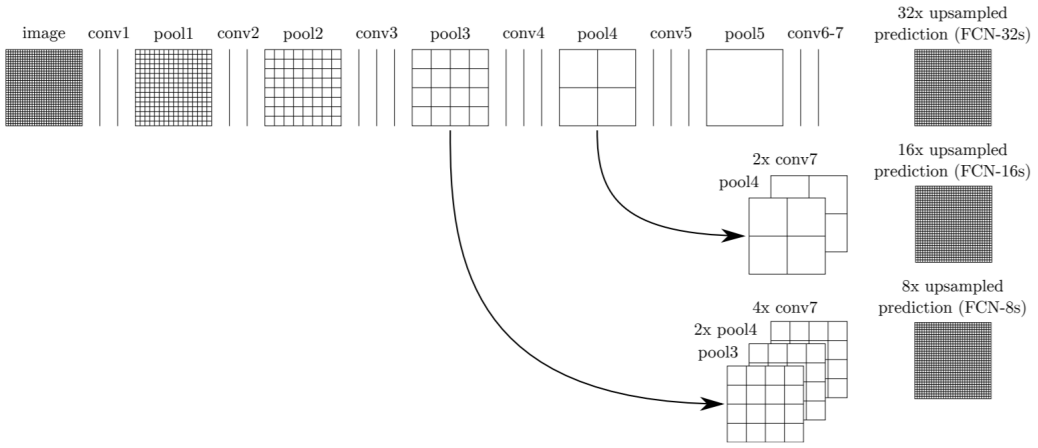

Figure 2.11: Upsampling structure.

However, although the results improved in the order of FCN8 than FCN 32, there were still many problems of missing details. In other words, FCN reduces the resolution through multiple layers of convolution and pooling and restores this reduced resolution through upsampling, so the details are either lost or overly smoothed, negatively affecting results.

To resolve the FCN issue, Noh *et al.* [38] proposed a deconvolution network composed of deconvolution and unpooling layers, which can identify pixelwise class labels and predict segmentation masks. **Figure 2.12** illustrates the detailed configuration of the entire deep network for segmentation.

The researchers deployed unpooling layers in the deconvolution network, which perform the reverse operation of pooling and reconstruct the original sizes of activations. As the output of the pooling layer is a sparse activa-

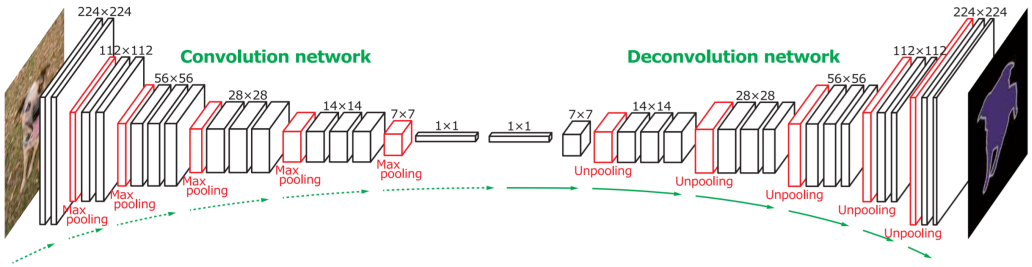

Figure 2.12: Deconvolutional network for semantic segmentation.

tion map, to recover the information of the feature maps they densified it through convolution-like operations with multiple learned filters. The overall operation is illustrated in **Figure 2.13**.

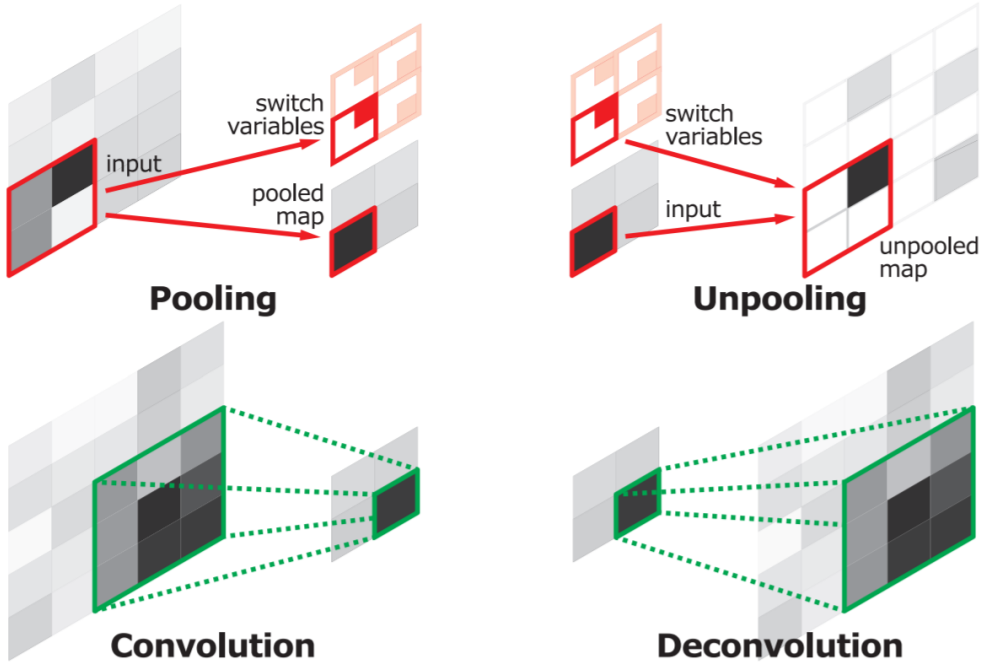

Figure 2.13: Unpooling and deconvolution.

2.4 Compressed deep learning

As deep learning has developed, there has been a corresponding disadvantage in that the number of training parameters had to increase to improve performance. Therefore, it is necessary to study small models with the similar performance. The advantages of a small model are as follows. First, the memory required for training can be reduced, and it can be easily applied to low-performance devices such as mobile devices. In addition, a high efficiency can be achieved when performing parallel training, and in the case of a system that needs to communicate with a server in real time, such as autonomous driving, the server is less overloaded and more frequent updates are possible.

SqueezeNet [21] proposed a Fire module, a strategy that allows a network to have as few parameters as possible. It employs three main strategies during the design of a network. First, it replaces 3×3 filters with 1×1 filters, as a 1×1 filter has nine times fewer parameters than a 3×3 filter. Second, it reduces the number of input channels to 3×3 because the total quantity of parameters in the layer is (number of input channels) $\times$ (number of filters) $\times$ (kernel size). Lastly, it performs the downsample as late as possible to secure several activation maps. The Fire module maintained performance despite reducing the number of parameters using two sizes of convolution in the layer, as illustrated in **Figure 2.14**.

Through this method, SqueezeNet, which has similar performance to AlexNet, reduces the number of parameters by 50 times, and has a model size of only 0.5 MB, was proposed.

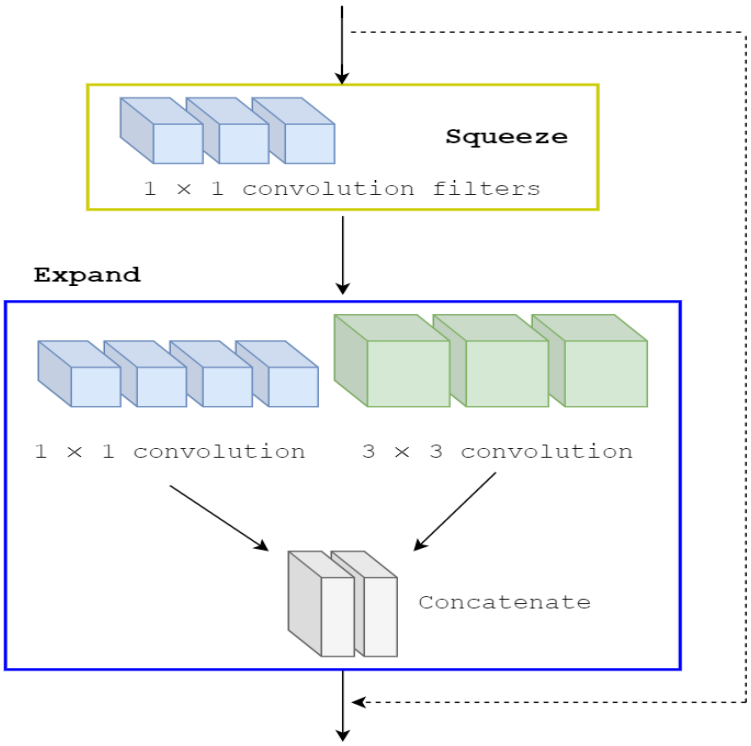

Figure 2.14: Organization of convolution filters in the Fire module.

Chapter 3

Fire Detection and Localization

In this chapter, we discuss fire detection and localization. As the terminology can be confusing, fire detection is the same as image classification and localization is the same as object detection in deep learning.

3.1 Related work

Recently, as the work in the field of deep learning has been increasing, it has been widely applied to studies in fire detection. In 2018, Khan *et al.* [34] proposed a fire detection network using GoogLeNet, and in 2019, Saeed *et al.* [46] slightly modified AlexNet to detect fire in images. In 2019, Arpit *et al.* [22] proposed a lightweight model applicable to Raspberry mobile devices, and Khan *et al.* [35] also proposed a fire detection network that can be used in Internet of Things (IoT) environments using MobileNet v2.

The above-mentioned methods, like the existing deep-learning method, use only the image as input, whereas in 2019 Cai *et al.* [3] proposed a network of multi-feature input data to compensate for insufficient experimental data. To construct multi-feature input data, they used HSV (hue, saturation, value) color space, YC_bC_r[64] color space, and a Canny edge detector. **Figure 3.1** is an example of the RGB image input used in most existing networks and the multi-feature input used in the network proposed by Cai *et al.*.

In addition, to detect and localize fire areas, DeepQuest AI, a company that easily integrates and deploys artificial intelligence, proposed FireNet: a

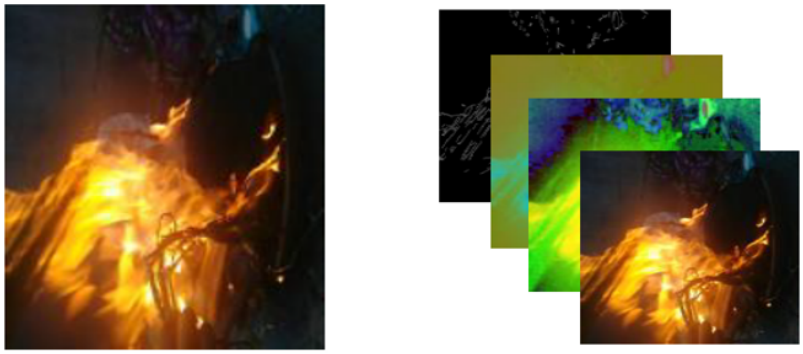

Figure 3.1: Left: RGB image input of existing networks. Right: multi-feature (Canny edge, YC_bC_r, HSV, RGB) input of the network proposed by Cai *et al.*

real-time fire detection project containing an annotated dataset with bounding-box information. The dataset consists of 502 images, split into 412 images for training and 90 images for testing. The network illustrated in **Figure 3.2** was a fine-tuned YOLO v3[43] that used darknet53 as the backbone network trained using the dataset.

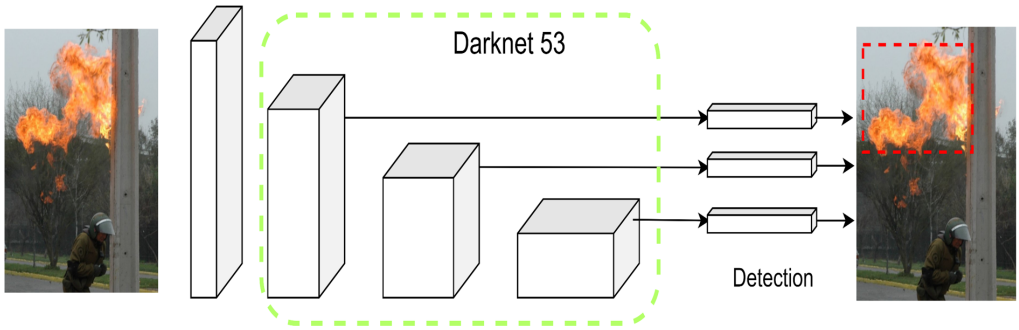

Figure 3.2: FireNet: YOLO v3.

3.2 Proposed method

We first propose an max pooling and fully connected (MFC) block to predict fire detection using feature maps of various sizes obtained through a convolution operation. The size of each feature map obtained for each stage is different, so a method of integrating feature maps into one form is necessary. The MFC block has 2×1 output fully connected from a feature map obtained through channel-wise max pooling. **Figure 3.3** is a brief description of the MFC block.

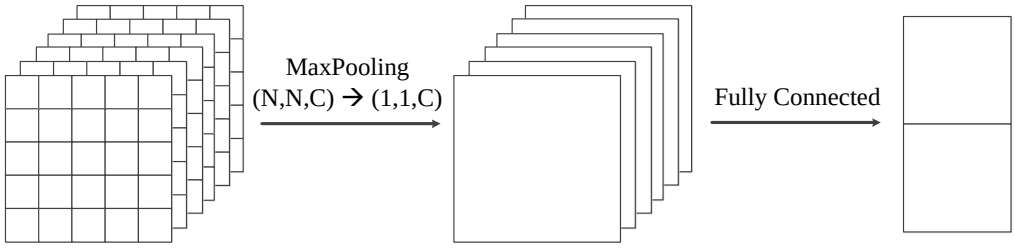

Figure 3.3: MFC block.

We constructed a network for fire detection with Res2Net [13] and ResNet blocks suitable for multi-scale representations. The Res2Net block proposed by Gao *et al.* represents multi-scale features at a granular level and increases the range of receptive fields for each network layer. It can also be easily applied to existing CNN models. In the Res2Net block, a Squeeze-and-Excitation (SE) block is used to recalibrate channel-wise feature responses by explicitly modelling interdependencies between channels [19]. **Figure 3.4** shows illustrations of the Res2Net and SE blocks.

First, in our network, the input shape of the architecture to be used for training was set to 224×224 and the depth was increased to 64 features using 3×3 convolution. After using the Res2Net block twice, downsampling was performed using stride-2 convolution. After one more iteration of this process, a feature map of size 28×28 was extracted from the feature map of size 56×56 using the ResNet instead of Res2Net block. A more detailed description of the architecture is given in **Figure 3.5** and **Table 3.1**.

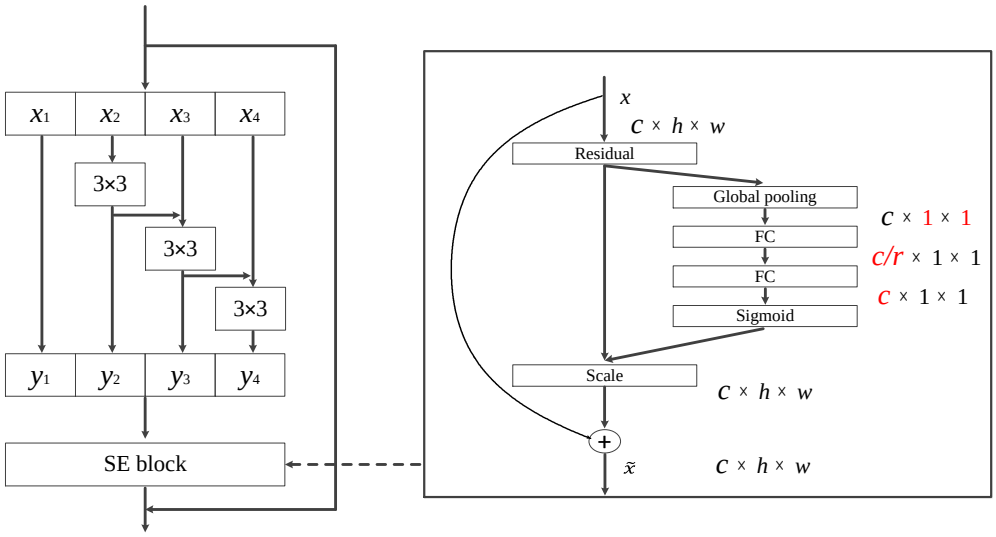

Figure 3.4: Res2Net and SE blocks.

Block type	Ingredients	Size of feature maps
inputs		$224 \times 224 \times 3$
entrance conv	convolution	$224 \times 224 \times 64$
downscaling1	Res2Net block + Res2Net block + conv(stride2)	$224 \times 224 \times 64$ $224 \times 224 \times 64$ $112 \times 112 \times 128$
downscaling2	Res2Net block + Res2Net block + conv(stride2)	$112 \times 112 \times 128$ $112 \times 112 \times 128$ $56 \times 56 \times 128$
downscaling3	ResNet block + ResNet block + conv(stride2)	$56 \times 56 \times 128$ $56 \times 56 \times 128$ $28 \times 28 \times 128$
downscaling4	ResNet block + ResNet block	$28 \times 28 \times 128$ $28 \times 28 \times 128$
output	MFC block	2×1

Table 3.1: Summary of our baseline architecture.

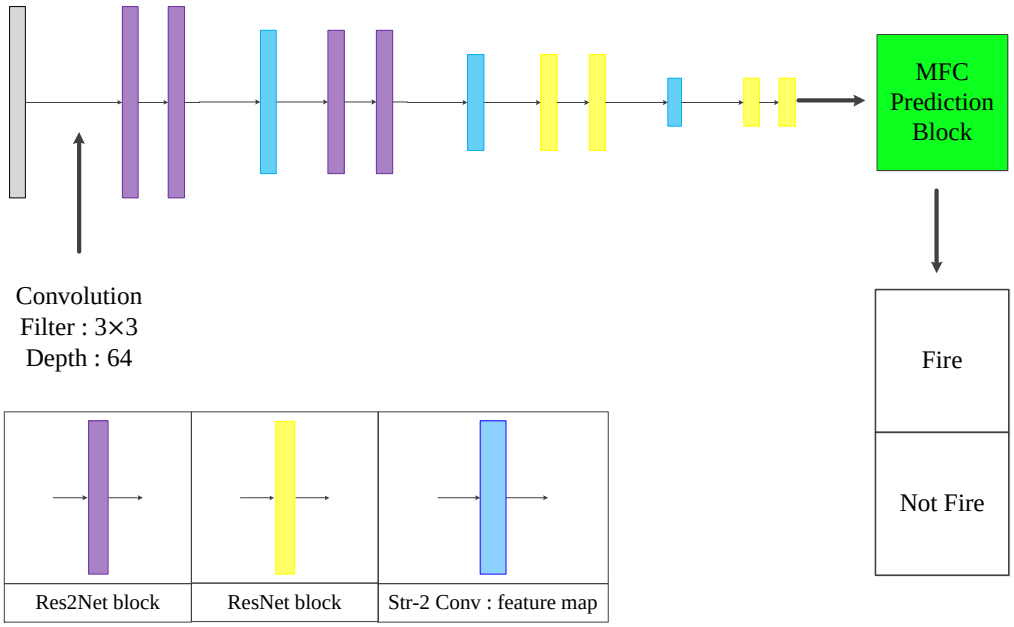

Figure 3.5: Fire detection model.

3.3 Experiments

When training, the epoch was set to 50 and training was performed more effectively using learning rate scheduling. The batch size was set to 16 and the optimizer used Adam, which combines AdaGrad and RMSProp. For the loss function, the binary cross-entropy function shown in **3.3.1** was used.

$$CE = -\sum_{i=1}^{C=2} t_i log(s_i) = -t_1 log(s_1) - (1 - t_1) log(1 - s_1) \tag{3.3.1}$$

To verify the validity of our network, we experimented with the RESCURE, BoWFire, Sharma, and Still datasets. The data used for the test comprise 1,885 images in total, and the detailed dataset configuration and description are as follows.

The RESCURE dataset [57] was created from a project that aimed to develop an interoperable information system that supports emergency and

crisis management command centers, enabling them to manage emergencies by quickly handling emergency situations and analyzing crisis crowdsourcing information as open data. As the RESCURE dataset is composed of videos, images were cut every 20 frames to avoid overlapping images and were used in the experiment. Through this, 1007 fire images and 406 non-fire images were generated. **Figure 3.6** shows example images of the RESCURE dataset.

Figure 3.6: RESCURE dataset: fire (left) and non-fire (right) images.

The BoWFire dataset was written by Chino *et al.* [7] in 2015. It consists of 226 images of various resolutions, divided into two categories: 119 containing fire and 107 without fire. Fire images consist of emergency situations involving various fire incidents, such as buildings on fire, industrial fires, car accidents, and riots. The rest of the images consist of emergency situations with no visible fire as well as images with fire-like regions, such as sunsets or red or yellow objects. **Figure 3.7** shows example images of the BoWFire dataset.

The Sharma dataset was created by Sharma *et al.* [48] in 2017. The dataset consists of 110 fire images and 541 non-fire images, which were collected from images on the internet. The non-fire image set was constructed to create a robust dataset for fire classification by constructing many images that looked similar to fire situations. The validity of the data was verified using networks such as VGG-16 and ResNet50. **Figure 3.8** shows example images of the Sharma dataset.

Figure 3.7: BoWFire dataset: fire (left) and non-fire (right) images.

Figure 3.8: Sharma dataset: fire (left) and non-fire (right) images.

The Still dataset was created by Mlích [33] in 2019. This dataset consists of 1,669 fire images and 4,581 non-fire images. **Figure 3.9** shows example images of the Still dataset.

To achieve more effective results using the data mentioned above, data augmentation was performed on the rest of the data except for the still dataset here. First, vertical-flip augmentation was used to minimize the loss of flame properties and random brightness was adjusted through gamma correction. **Figure 3.10** is an example of an image with augmentation applied.

Each dataset was divided into a training set and a test set in a ratio

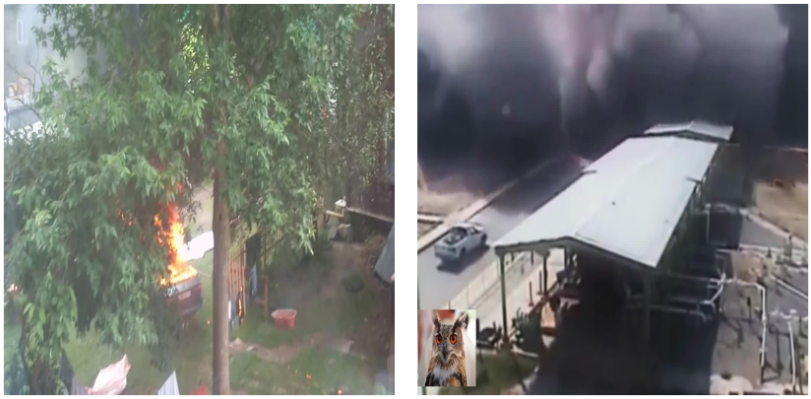

Figure 3.9: still_image dataset: fire (left) and non-fire (right) images.

Figure 3.10: Example of data augmentation.

of 80:20, and the data augmentation mentioned above was performed on the training set. Through this, 12,596 images for the entire training set and 1,885 for the test set were used.

To emphasize the performance of our proposed method, the recently published fire detection networks proposed by Khan *et al.*, Saeed *et al.*, and Arpit *et al.* were used. We used the accuracy, precision, recall, F1 scores, and false alarm rate (FAR) widely used in fire detection as evaluation metrics to confirm the validity of the experimental results. Each evaluation metric is as shown in equation **3.3.2**. In **3.3.2**, T_P means true positive, T_N means true negative, F_P means false positive, and F_N means false negative.

$$\begin{aligned} \text{Accuracy} &= \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \\ \text{Precision} &= \frac{T_P}{T_P + F_P} \\ \text{Recall} &= \frac{T_P}{T_P + F_N} \\ \text{F1score} &= 2 \times \frac{Precision \times Recall}{Precision + Recall} \\ \text{FAR} &= \frac{F_P}{T_N + F_P} \times 100 \end{aligned} \tag{3.3.2}$$

Table 3.2 to **Table 3.6** are the results of each of the fire detection evaluation metrics. As mentioned earlier, the total number of images used in the test was 1,885, which consisted of 283 from the RESCUE dataset, 91 from the BoWFire dataset, 261 from the Sharma dataset, and 1,250 from the Still dataset. At most values, the performance of our proposed model is better than other models. Especially in the entire data, the results of all evaluation metrics are higher than others.

Method	Entire	RESCURE	BoWFire	Sharma	Still
Khan (GoogleNet)	0.9437	0.9611	0.8571	0.9655	0.9416
Saeed (AlexNet)	0.9385	0.9575	0.8791	0.9386	0.8276
Arpit (FireNet)	0.8997	0.9434	0.8461	0.9386	0.8856
Khan(MobileNetv2)	0.9326	0.9717	0.8791	0.9501	0.9280
Khan (VGGNet)	0.9129	0.8975	0.8241	0.9425	0.9168
Proposed	**0.9623**	**0.9788**	**0.9780**	**0.9693**	**0.956**

Table 3.2: Accuracy results.

Method	Entire	RESCURE	BoWFire	Sharma	Still
Khan (GoogleNet)	0.9399	0.9653	0.9137	**0.9423**	**0.9287**
Saeed (AlexNet)	0.9317	**0.9845**	0.8793	0.9038	0.9141
Arpit (FireNet)	0.8772	0.9692	0.9056	0.8571	0.8184
Khan (MobileNetv2)	0.9131	0.9751	0.9791	0.8518	0.8798
Khan (VGGNet)	0.9050	0.9521	0.8867	0.8600	0.8867
Proposed	**0.9502**	0.9801	**0.9824**	0.9387	0.9285

Table 3.3: Precision results.

Method	Entire	RESCURE	BoWFire	Sharma	Still
Khan (GoogleNet)	0.9384	0.9798	0.9298	**0.9607**	**0.9125**
Saeed (AlexNet)	0.9030	0.9597	0.8947	0.9215	0.8688
Arpit (FireNet)	0.8246	0.9497	0.8421	0.8235	0.7492
Khan (MobileNetv2)	0.8892	0.9849	0.8245	0.9019	0.8542
Khan (VGGNet)	0.8353	0.8994	0.8245	0.8431	0.7988
Proposed	**0.9400**	**0.9899**	**0.9824**	0.9019	0.9096

Table 3.4: Recall results.

Method	Entire	RESCURE	BoWFire	Sharma	Still
Khan (GoogleNet)	0.9391	0.9725	0.9217	**0.9514**	**0.9205**
Saeed (AlexNet)	0.9171	0.9720	0.8869	0.9126	0.8908
Arpit (FireNet)	0.8501	0.9593	0.8727	0.8400	0.7823
Khan (MobileNetv2)	0.9010	0.9800	0.8952	0.8761	0.8668
Khan (VGGNet)	0.8688	0.9250	0.8545	0.8514	0.8404
Proposed	**0.9450**	**0.9850**	**0.9824**	0.9200	0.9190

Table 3.5: F1-score results.

Method	Entire	RESCURE	BoWFire	Sharma	Still
Khan (GoogleNet)	3.1578	8.3333	14.7058	**1.4285**	**2.6460**
Saeed (AlexNet)	3.4817	**3.5714**	20.5882	2.3809	3.0871
Arpit (FireNet)	6.0728	7.1428	14.7058	3.3333	6.2844
Khan (MobileNetv2)	4.4534	5.9523	**2.9411**	3.8095	4.4101
Khan (VGGNet)	4.6153	10.7142	17.6470	3.3333	3.8588
Proposed	**2.5910**	4.7619	**2.9411**	**1.4285**	**2.6460**

Table 3.6: False alarm rate results.

We also conducted additional experiments with fire blocking and noise attacks to confirm the robustness of the proposed model. As in Khan *et al.* (MobileNet v2), it was confirmed that fire detection does not occur outside the fire region by covering the fire region with the background in the image or adding white and red blocks in the fire image. In addition, it was confirmed that fire detection is also good when additive white Gaussian noise (AWGN) is applied to the fire area and scratch is artificially added to the fire region. In **Figure 3.11**, we can see that in each case our model predicts properly and the probability does not change.

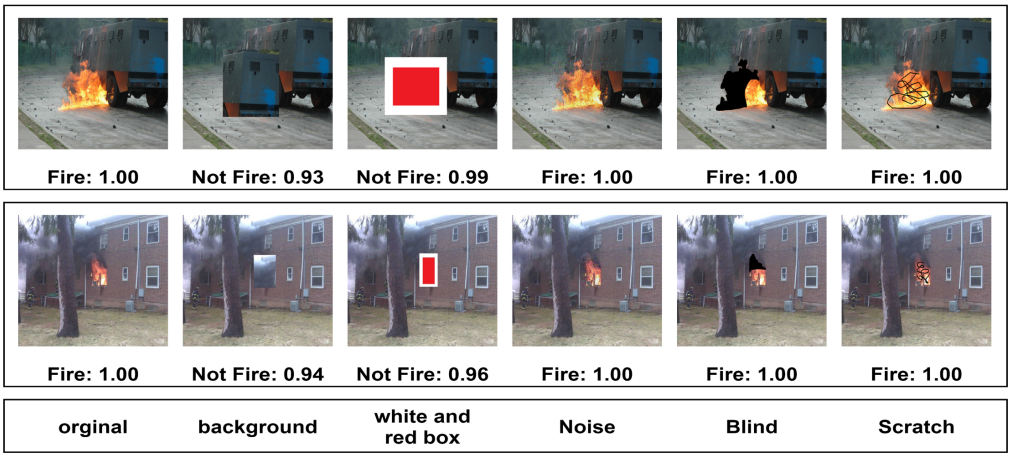

Figure 3.11: Robustness analysis of the proposed fire detection model.

3.3.1 Fire area localization

A Class Activation Map (CAM) [63], proposed by Zhou *et al.*, is one of the methods to interpret CNN. It is possible to analyze which parts of the image have most significantly influenced the results of the model. However, it has the disadvantage of having to change the fully connected layer of the network to global average pooling and fine-tuning the model after changing it. To compensate for these shortcomings, Selvaraju *et al.* proposed Grad-CAM [47], a generalized version of CAM that does not need to replace these fully connected layers with global average pooling. In other words, through Grad-CAM, the model can recognize the predictable information through any part of the image, so it can localize to the position of the object.

Through this Grad-CAM, bounding-box information can be obtained freely. In addition, as in the existing object detection network, if NMS is applied to remove the duplicated bounding box, plausible bounding-box information can also be obtained. **Figure 3.12** shows the result of Grad-CAM and localization of the fire area using this method. The results of Grad-CAM are shown in the middle of **Figure 3.12**; it is classified as fire by focusing on the fire area and it is localized by a given specific thresholding value in **Figure 3.12** on the right of the second row.

Localization through Grad-CAM has the advantage that it can be obtained without further training, but the performance of the result is poor, as shown in the first row of **Figure 3.12**. To improve the localization performance, the object detection method mentioned before should be applied. Here, we localize the fire area using Faster R-CNN and YOLO v3. To train the two networks, we used the data provided by DeepQuest AI, which include bounding-box information and are categorized as small and large fire areas. **Figure 3.13** shows examples of the data. The Still dataset was used to provide test images for fire detection using the two networks, and the results are shown in **Table 3.7**.

As shown in **Table 3.7**, neither network performs well in fire detection. In particular, in the case of YOLO v3, the precision is high but the recall rate is low, which means that the fire detection performance is poor. In the case of Faster R-CNN, the recall is high but the precision is low, so it finds fire areas well, but it can be seen that it even finds fire areas in the non-fire

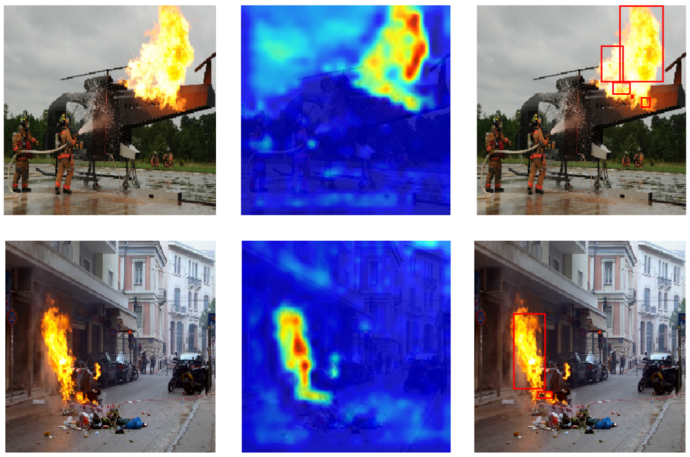

Figure 3.12: Input image (left), Grad-CAM (middle), and bounding boxes (right).

	Still images			
Method	Fire	Not fire	Recall	Precision
YOLO v3	1,072	4,275	0.642	**0.777**
Faster R-CNN	1,491	3,118	**0.893**	0.504

Table 3.7: Fire detection results from YOLO v3 and Faster R-CNN.

images. In the case of YOLO v3, the problem is that the detection performance itself needs to be increased, but for Faster R-CNN, the disadvantages can be compensated by using the fire detection network proposed above.

By first classifying the image through the fire detection network and localizing the image classified as a fire image using Faster R-CNN, its disadvantages can be compensated. As shown in **Table 3.8**, the precision of Faster R-CNN increases and the recall decreases slightly when fire detection proceeds in two steps.

Figure 3.13: DeepQuest AI data: Large fire (left), small fire (right).

	Still Images			
Method	Fire	Not fire	Recall	Precision
YOLO v3	1,061	4,573	0.635	**0.992**
Faster R-CNN	1,464	4,560	**0.877**	0.985

Table 3.8: Two-step fire detection results (fire detection network and then YOLO v3 or Faster R-CNN).

3.3.2 Fire localization results

The average precision (AP) proposed by PASCAL VOC [11] in 2010 is used as a metric to evaluate this localization. AP is calculated using Intersection over Union (IoU), precision, and recall. IoU is a measurement for determining the overlap between two areas. It requires a ground-truth bounding box and a predicted bounding box. IoU is given by the overlapping area between the predicted bounding box and the ground-truth bounding box divided by the area of union between them; **Figure 3.14** illustrates the IoU between a ground-truth bounding box (in green) and a detected bounding box (in red). T_P and F_P are calculated by a specific IoU value and probability, and and precision and recall are calculated as in the equation introduced in the previous classification metrics. Based on the precision and recall, the AP is calculated in the same way as **3.3.3**.

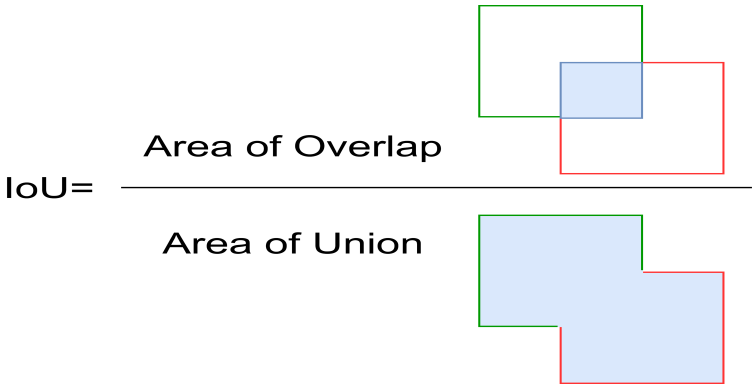

Figure 3.14: Computing the Intersection over Union (IoU).

$$\begin{aligned} & \sum_{n=0} (r_{n+1} - r_n) \rho_{interp}(r_{n+1}) \\ \text{with} \quad & \rho_{interp}(r_{n+1}) = \max_{\tilde{r}:\tilde{r} \geq r_{n+1}} \rho(\tilde{r}) \end{aligned}, \tag{3.3.3}$$

where $\rho(\tilde{r})$ is the measured precision at recall $\tilde{r}$.

Figure 3.15 shows the precision $\times$ recall curve for AP_{50} values of two fire localization networks. We tested 90 validated images of the FireNet dataset and calculated precision and recall based on the IoU value of 0.5. According to **3.3.3**, the area under the curve becomes the AP value. **Figure 3.16** shows examples of the localization results of the two models for the Still image dataset; the blue boxes are misclassified and the green is properly classified. YOLO v3 is often unable to ascertain the fire area compared to Faster R-CNN.

Table 3.9 shows the results of AP measurement based on various IoUs, similar to existing object detection methods. In **Table 3.9**, AP is calculated by the average of AP_{50} to AP_{95} in five steps, as used in the COCO metric, and AP^{large} is the AP except for the small fire area. Grad-CAM shows reasonable results only in AP_{10} because the bounding results are not combined into one but divided into several, as shown in the right figure of the first row of **Figure 3.12**. As mentioned earlier, the results of Faster R-CNN show higher performance than YOLO v3 in all AP metrics.

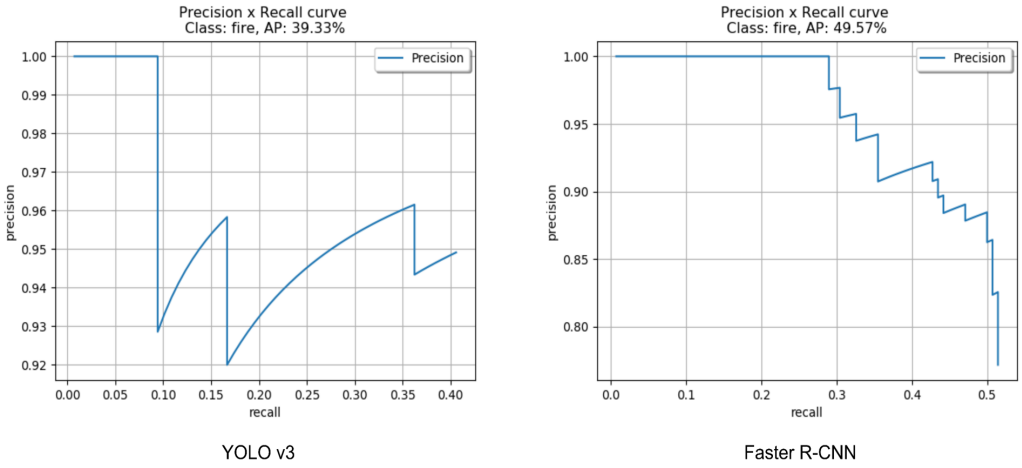

Figure 3.15: AP_{50} results of YOLO v3 (left) and Faster R-CNN (right).

	AP_{10}	AP_{50}	AP_{75}	AP	AP^{large}
Grad-CAM	13.18	3.64	0.03	1.06	1.645
YOLO v3	40.22	39.33	23.14	22.32	29.51
Faster R-CNN	**50.17**	**49.57**	**33.66**	**30.68**	**46.58**

Table 3.9: AP results according to changes in IoU.

3.4 Conclusion

In this chapter, we have proposed a fire detection network and confirmed that it is superior to existing networks from various datasets. Through additional experiments, it was found that good prediction results were obtained even when attacked in various environments. Moreover, the fire detection performance of the localization model was improved by using the fire detection model before the localization model.

Figure 3.16: Examples of the localization results of YOLO v3 and Faster R-CNN for the Still dataset.

Chapter 4

Semantic Segmentation using Deep Learning for Fire Images

In this chapter, we propose a fire segmentation network using deep learning and compare it with existing binary segmentation networks and image-processing algorithms for the FiSmo dataset and Corsican Fire Database. We will also analyze why it works better than existing networks using deep learning.

4.1 Related work

Celik *et al.* [64] proposed a fire detection algorithm based on components of the $\mathrm{YC}_b\mathrm{C}_r$ color space for classifying flame pixels. In this method, to separate the luminance and the color difference, the RGB color space is converted to the $\mathrm{YC}_b\mathrm{C}_r$ color space using **4.1.1**. In addition, **4.1.2** was defined to separate flame pixels by eliminating the harmful effects of varying illuminance.

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.2568 & 0.5041 & 0.0979 \\ -0.1482 & -0.2910 & 0.4392 \\ 0.4392 & -0.3678 & -0.0714 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \tag{4.1.1}$$

Here, Y is luminance, C_b and C_r denote chrominance, τ is a constant determined by analysis of ROC (receiver operating characteristics), and a pixel fulfilling $F(x, y) = 1$ is defined as a flame pixel.

$$F(x,y) = \begin{cases} 1 & \text{if } Y > Y_{mean}, C_b < C_{b_{mean}}, \\ & C_r > C_{r_{mean}}, |C_r - C_b| \ > \tau \\ 0 & \text{otherwise} \end{cases} \tag{4.1.2}$$

Because flames vary considerably depending on the type of material to be burned, the surrounding environment, and the characteristics of the camera, there are many problems with detecting flame areas using only color information. To cope with these problems, Tuba *et al.* [54] achieved higher accuracy by adding the following constraints **4.1.3**, **4.1.4**, and **4.1.5** to the flame classification condition.

$$F_1(x,y) = \begin{cases} 1 & \text{if } Y \geq C_b \\ 0 & \text{otherwise} \end{cases} \tag{4.1.3}$$

$$F_2(x,y) = \begin{cases} 1 & \text{if } C_r \geq C_b \\ 0 & \text{otherwise} \end{cases} \tag{4.1.4}$$

$$F_3(x,y) = \begin{cases} 1 & \text{if } C_b \leq Th_b, C_r \geq Th_r \\ 0 & \text{otherwise} \end{cases} \tag{4.1.5}$$

This color model has a fast fire detection speed, but still has the disadvantages that the false detection rate is high when there is an object of a color similar to a flame and that there is a limit affected by the illumination

of the camera and the environment.

Töreyin *et al.* [55] proposed a conventional fire detection algorithm based on image processing, which uses the frequency of flames and a method using flame movements. Although there are differences in the material of the flame and the wind, the color or brightness of a flame image pixel varies at a frequency of about 10 Hz. Fire is detected through wavelet transformation using the frequency of the flame, or flicker. In addition, Yamagishi *et al.* [60] proposed a method using the Hue, Saturation, and Value (HSV) color space, and Kim *et al.* [24] proposed an algorithm for using an infrared image to detect the flame area.

As mentioned before, since AlexNet, studies on deep-learning architecture have been actively conducted, and methods using deep learning have been applied to various fields. For the image segmentation task, attempts have been made to apply the architecture of the existing deep-learning model [16]; however, the geometric information of the image is lost as it passes through the fully connected layer.

To address these problems, Long *et al.* [31] proposed a deep-learning model based on FCN without a fully connected layer. This led to the creation of a model composed entirely of the convolution. FCN improves the effectiveness of semantic image segmentation by preserving geometric information. However, the prediction of FCN is fragmented according to the size of the object, and the region detail of the object cannot be effectively expressed. In 2015, Noh *et al.* [38] solved this problem by using deconvolution for the upsampling method.

In 2015, Ronneberger *et al.* proposed U-Net [45] by integrating a skip connection into the deconvolutional FCN model. U-Net was examined based on warping error, rand error, and pixel error index in the electron microscopy (EM) segmentation challenge using FCN, skip connection, and feature concatenation, and outperformed the previous best methods.

FusionNet [41] was proposed in 2016 by Quan *et al.* and is a model for binary segmentation in EM images. The model was an improved network developed from U-Net based on FCN, and used encoding and decoding to reduce the amount of computation. It also compensated for the disadvantages of U-Net by using residual blocks and summation-based skip connections. Fu-

sionNet outperformed existing segmentation methods, including U-Net, for the ISBI2012 EM segmentation challenge data.

Drozdzal *et al.* [10] proposed the FC-ResNet model for medical images that addresses the task of image segmentation using short skip connection, long skip connect, and various concepts of bottleneck block. In 2018, they developed the model and proposed FCN+FC-ResNet [9] by combining FCN and FC-ResNet. They confirmed excellent performance of their model for EM, liver lesion, and prostate datasets.

In 2019, Yuan *et al.* [61] proposed a deep smoke segmentation model for detecting smoke areas that divide into a coarse path and a fine path. The first path extracts the global context information of smoke and the second path obtains its fine spatial details. They achieved better performance than recently published segmentation methods for smoke datasets.

4.2 Proposed architecture

Block type	Ingredients	Size of feature maps
inputs		$640 \times 640 \times 3$
entrance conv	convolution	$640 \times 640 \times 64$
downscaling1	conv+res+conv +merge +conv(stride-2)	$640 \times 640 \times 64$ $320 \times 320 \times 64$
downscaling2	conv+res+conv +merge +conv(stride-2)	$320 \times 320 \times 128$ $160 \times 160 \times 128$
downscaling3	conv+res+conv +merge +conv(stride-2)	$160 \times 160 \times 256$ $80 \times 80 \times 256$
downscaling4	conv+res+conv +merge +conv(stride-2)	$80 \times 80 \times 512$ $40 \times 40 \times 512$
bridge	conv+res +conv+merge	$40 \times 40 \times 1024$ $40 \times 40 \times 512$
upscaling4	deconv+conv+res +conv+merge	$80 \times 80 \times 512$ $80 \times 80 \times 256$
upscaling3	deconv+conv+res +conv+merge	$160 \times 160 \times 256$ $160 \times 160 \times 128$
upscaling2	deconv+conv+res +conv+merge	$320 \times 320 \times 128$ $320 \times 320 \times 64$
upscaling1	deconv+conv+res +conv+merge	$640 \times 640 \times 64$ $640 \times 640 \times 64$
exit conv	convolution	$640 \times 640 \times 1$

Table 4.1: Summary of proposed architecture.

In our proposed network, we set the input and output image size to 640×640, similarly to FusionNet. However, unlike FusionNet, this network deploys both the entrance and exit convolutions as it can adjust the number of feature maps. In addition, we designed the middle skip connection to compensate for errors that may occur in the existing architecture. We also modified FusionNet to increase the ensemble effect using the middle skip connection, as the residual block is referred to as the ensemble system by Veit *et al.* [56].

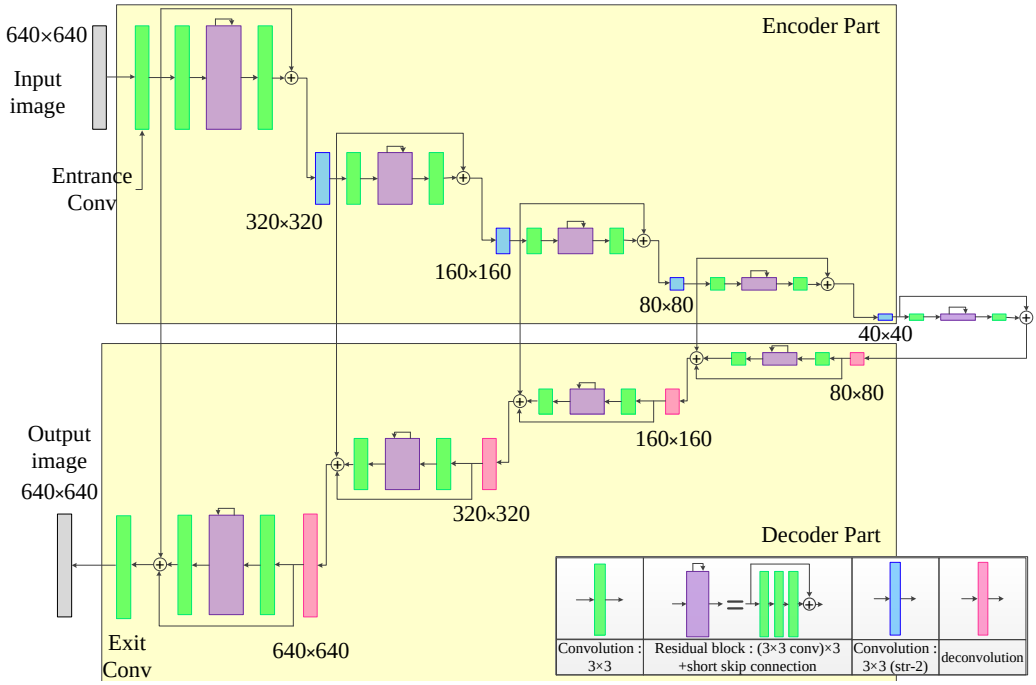

Figure 4.1: Proposed architecture.

In our network, the convolution and deconvolution filter sizes are unified to 3×3 and all the activation functions use the ReLU. The activation functions and batch normalization are performed in all blocks. A detailed description of the proposed architecture is divided into the encoder part and the decoder part.

The encoder part is a process of extracting features and proceeds as follows. After the input image is inserted into the entrance convolution, the result is used as three inputs: the next convolution block, the middle skip connection, and the long skip connection. The result obtained from the feature map passing through the convolution block, residual block, and convolution block (hereafter denoted as Conv+Res+Conv block) is then added to the feature map extracted from the entrance convolution. This addition process is defined as the middle skip connection.

Subsequently, without utilizing max pooling, as used in FusionNet, this study reduces the size of the feature map through stride-2 convolution. The

obtained feature map is deployed as three inputs as in the previous step. This method is repeated until the width and height of the feature map become 40×40.

The decoder part is the process of restoring the features extracted from the encoder to the original image size and proceeds as follows. The result of the encoder part passes through the Conv+Res+Conv block with the middle skip connection (bridge) and then the size of the feature map is expanded to 80×80 using deconvolution. The expanded feature map is used as the input for the next convolution and middle skip connection. The feature map passed through the Conv+Res+Conv block is combined with the middle skip connection and the above-mentioned feature map of the long skip connection.

We input the result obtained from the deconvolution of the feature map into the convolution block and the middle skip connection again. This process is repeated until the width and height of the feature map are 640×640, the size of the original image. Finally, we integrate all the feature maps of the convolution block, middle skip connection, and long skip connection; the result becomes depth 1 through the exit convolution.

A more detailed description of the architecture is given in **Figure 4.1** and **Table 4.1**. In **Figure 4.1**, the green block is the convolution block, the purple block is the residual block, the blue block is the stride-2 convolution block, and the red block is the deconvolution block. Similarly to FusionNet, the residual block is composed of three convolution blocks and the short skip connection.

4.2.1 Comparison with FusionNet

The architecture proposed in this chapter is similar to FusionNet. However, the entrance convolution, exit convolution, and middle skip connection are added in this network. The entrance convolution is designed to match the output shape of Conv+Res+Conv equally to the middle skip connection. The exit convolution is designed so that the depth of the final output image is 1. Lastly, the reason for suggesting the middle skip connection is as follows.

Here, FusionNet and the proposed middle skip connection effect can be expressed as **4.2.1** as shown in **Figure 4.2**.

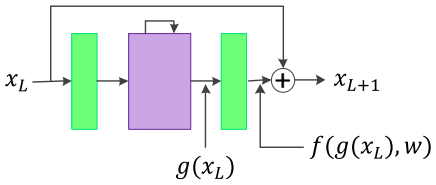

Figure 4.2: One stage of the Conv+Res+Conv part.

$$x_{L+1} = \begin{cases} f(g(x_L), W) & \text{FusionNet} \\ x_L + f(g(x_L), W) & \text{Proposed method} \end{cases} \tag{4.2.1}$$

According to [17], denoting the loss function as $\mathcal{E}$, we can obtain the following **4.2.2** from the chain rule of backpropagation.

$$\begin{aligned} \frac{\partial \mathcal{E}}{\partial x_L} &= \frac{\partial \mathcal{E}}{\partial x_{L+1}} \frac{\partial x_{L+1}}{\partial x_L} \\ &= \begin{cases} \frac{\partial \mathcal{E}}{\partial x_{L+1}} \frac{\partial}{\partial x_L} f(g(x_L), W) & \text{FusionNet} \\ \frac{\partial \mathcal{E}}{\partial x_{L+1}} (1 + \frac{\partial}{\partial x_L} f(g(x_L), W)) & \text{Proposed method} \end{cases} \end{aligned} \tag{4.2.2}$$

As the term $f(g(x_L), W)$ of FusionNet in **4.2.2** has the activation function as ReLU, $\frac{\partial}{\partial x_L} f(g(x_L), W)$ can be zero. However, as mentioned in [17], in general the term of $\frac{\partial}{\partial x_L} f(g(x_L), W)$ in the proposed method in **4.2.2** cannot be always -1 for all samples in a mini-batch. This shows that the method proposed in this chapter can more effectively avoid vanishing gradients compared to FusionNet.

4.3 Experimental results

In this section, we conducted various experiments with the FiSmo dataset and Corsican Fire Database to verify our proposed model. **Figure 4.3** shows examples of the FiSmo dataset and Corsican Fire Database, and we explain the data in detail in the next subsections. The data used for the experiment

were divided into 80% and 20% for the training and test data, respectively.

When training, the mean square error (MSE) and Adam optimizer were used as the loss function and optimizer, respectively. The epoch and batch size were set to 100 and 2, respectively.

Figure 4.3: FiSmo dataset and Corsican Fire Database examples.

Before comparing the ground-truth image and predicted image, the binary process regards a predicted value greater than 0.5 as a fire with label "1", and a value less than 0.5 as background with label "0". To confirm the similarity between the ground-truth image and the predicted result image, the IoU mentioned in the localization was deployed as an evaluation metric.

We also compared the proposed model with FusionNet, FCN+FC-ResNet, Color YC_bC_r feature algorithm, and Deep Smoke. Through experimentation, it was confirmed that Deep Smoke resulted in a generally smooth prediction form, and Color YC_bC_r resulted in generally fragmented results.

4.3.1 Experimental results using FiSmo dataset

The FiSmo dataset, consisting of image and video data, was made by Cazzolato *et al.* in 2017. It is composed of four categories: Flickr-FireSmoke,

Flickr-Fire, BoWFire, and SmokeBlock. Each dataset is composed as follows: the Flickr-FireSmoke dataset contains 5556 images classified as fire and smoke; Flickr-Fire is composed of 2000 images, 1000 labeled as "fire" and the remainder labeled as "not fire"; The BoWFire dataset consists of 226 images, 119 labeled as "fire" and 107 labeled as "not fire"; and SmokeBlock is composed of 1666 images, 832 labeled as "smoke" and 834 labeled as "not smoke".

The image data required in this study are the fire image and its segmentation mask image that is divided in to fire and background. In this study, we use only the BoWFire dataset, as it provides the fire image and its segmentation mask image. Therefore, 118 data with fire and segmentation mask pairs were used in the BoWFire subset, except for one image in which the ground-truth image is unclear.

As the dataset is small, to increase the training effect, we performed data augmentation wherein only the left and right symmetry may be modified without having a significant effect, as shown in **Figure 4.4**. As a result, the total number of data used for training and testing was 236.

Figure 4.4: Example of FiSmo image augmentation.

The training parameters are as follows: The total iteration number obtained is approximately 18,000. The learning rate is 10^{-2} by 1000 iterations,

10^{-3} by 3000 iterations, 10^{-4} by 5000 iterations, 10^{-5} by 10,000 iterations, 5×10^{-6} by 15,000 iterations, and 10^{-6} beyond. The results obtained through this method are presented subsequently.

The first row of **Figure 4.5** shows the results for the fire097.png image. In FusionNet and Color YC_bC_r, as indicated by the yellow arrow in the lower left, an orange lane was misclassified as fire because of its color. Furthermore, in the FCN+FC-ResNet, as indicated by the red circle in the upper right, some details were lost. However, using the proposed architecture, the fire area stands out distinctly.

The second row in **Figure 4.5** is the result for the fire104.png image. In FCN+FC-ResNet and Color YC_bC_r, an orange lane at the bottom center of the image was misclassified as fire because of its color. In FusionNet, as indicated by the red circle in the same position, it is also misclassified due to color. The proposed method can accurately identify the fire area, unlike the other methods.

The third row of **Figure 4.5** is the result for the fire105.png image. Except for the FCN+FC ResNet, the fire area was classified well. Unlike other models, our network even classified the pillars in the middle of the fire.

The fourth row in **Figure 4.5** is the result for the fire106.png image. In FusionNet and Color YC_bC_r, as indicated by the red circle in the middle, some details were lost. Furthermore, the FCN+FC-ResNet has considerable misclassification of fire segmentation. The proposed method can accurately identify the fire area, unlike the other methods.

The fifth row of **Figure 4.5** is the result for the fire112.png image. As indicated by the yellow arrow, FusionNet misclassified the orange object.

The sixth row of **Figure 4.5** is the result for the fire115.png image. As indicated by the yellow arrow, FusionNet and FCN+FC-ResNet have the disadvantage of classifying the building near the fire as fire. The proposed method can accurately identify the fire area, unlike the other methods. The area marked with a yellow circle is a false positive that is not visible in the ground-truth image, but is recognizable as fire in the input image.

The average values of the IoU obtained for the entire FiSmo dataset are shown in **Table 4.2**. Through this, it was found that the proposed method outperforms the other methods for the FiSmo dataset.

Figure 4.5: FiSmo image results. The order of the figure is 97, 104, 105, 106, 112, 115.

4.3.2 Experimental results using Corsican Fire Database

The Corsican Fire Database, created by Toulouse *et al.*, consists of fire images shot outdoors in 2017. The dataset consists of images of forest that have been reported in Corsica. their includes RGB images, near-infrared images, and their corresponding ground-truth images, which are black-and-white images with the fire segmented regions created by experts. All the images and their corresponding template images are in .png format.

This dataset consists of 595 images, of which more than 90 have near-infrared spectrum images. The dataset also contains five 540-frame videos

Method	IoU
FusionNet	0.63653529
FCN+FC-ResNet	0.66398529
YC_bC_r Color feature	0.47071706
Deep Smoke	0.59600632
Proposed method	**0.79145361**

Table 4.2: Comparison of results with other methods – FiSmo dataset.

shot from a fixed direction, and it also contains the near-infrared spectrum images.

In this study, segmentation was performed using RGB fire images generated in various places; thus, the video image data taken in a fixed direction and near-infrared spectrum images were excluded. Therefore, the number of training and test data is 595. The total iteration number obtained is approximately 47,000. The learning rate scheduling for training is as follows: 10^{-2} by 1000 iterations, 10^{-3} by 3000 iterations, 10^{-4} by 5000 iterations, 10^{-5} by 15,000 iterations, 5×10^{-6} by 20,000 iterations, and 10^{-6} beyond. The results obtained through this method are as follows.

The first row of **Figure 4.6** is the result for the 408_rgb.png image. For FusionNet, FCN+FC-ResNet, Deep Smoke, and YC_bC_r, some misclassified results can be identified around the fire. However, this problem is solved by using the proposed model.

The second row of **Figure 4.6** is the result for the 414_rgb.png image. For YC_bC_r, it is not properly classified. However, the proposed architecture yields better classification results.

The third and sixth rows of **Figure 4.6** are the results for the 422_rgb.png and 493_rgb.png images, respectively. For FusionNet and FCN+FC-ResNet, as indicated by the red circle, some details were lost or smoke was incorrectly classified as fire.

The fourth and fifth rows of **Figure 4.6** are the results for the 423_rgb.png and 434_rgb.png images, respectively. For FusionNet and FCN+FC-ResNet, as indicated by the red circle, the fireman's fire suit was misclassified as fire because of its color. However, the model we proposed classified the area, not

the fire.

The average values of the IoU for the Corsican Fire Database are shown in **Table 4.3**. Through this, it was found that the proposed method outperformed the other methods for the Corsican Fire Database.

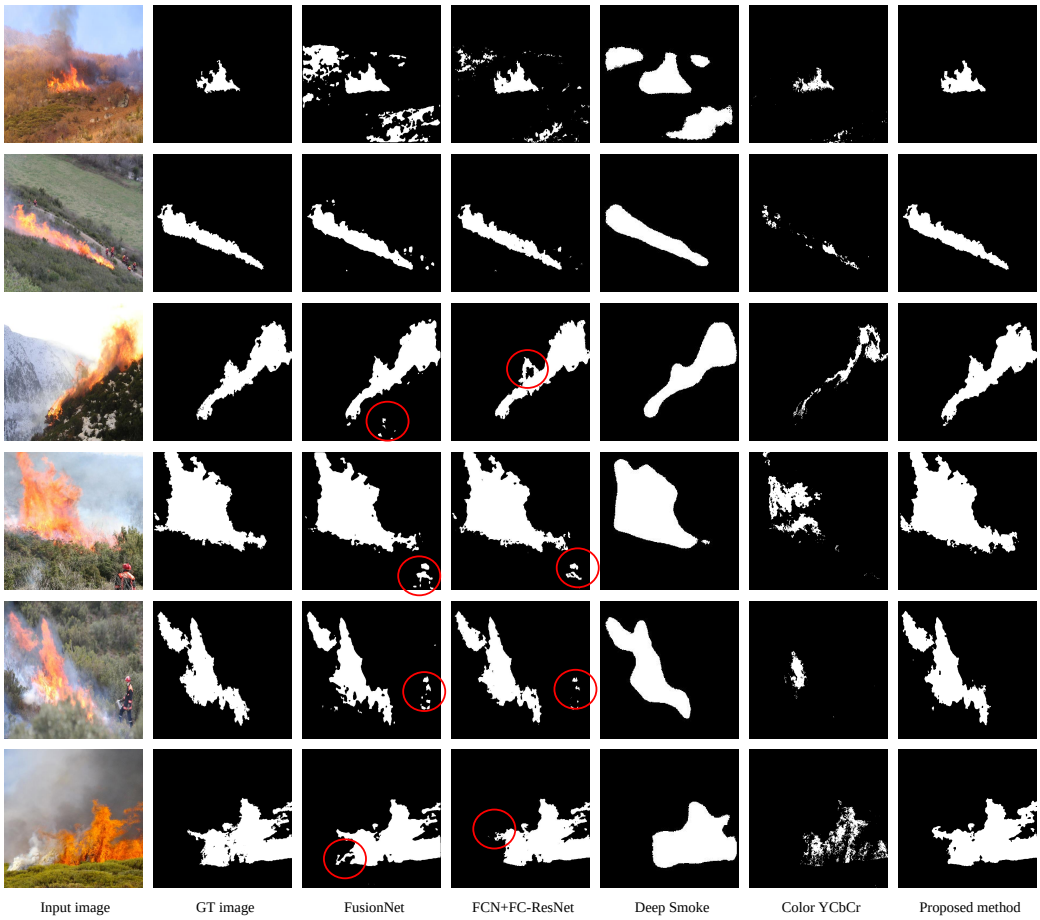

Figure 4.6: Corsican Fire Database image results. The order of the figure is 408, 414, 422, 423, 434, 493.

Method	IoU
FusionNet	0.77014998
FCN+FC-ResNet	0.85016260
YC_bC_r Color Feature	0.42878885
Deep Smoke	0.76479557
Proposed method	**0.90020717**

Table 4.3: Comparison of results with other methods – Corsican Fire Database.

4.4 Conclusion

In this chapter, we proposed a new deep-learning architecture based on FusionNet, which demonstrates excellent performance in the existing semantic image segmentation model. The proposed model, however, is more powerful, as it incorporates the novel concept of entrance convolution, exit convolution, and middle skip connection into the structure of FusionNet. In our experiments with the FiSmo and Corsican datasets, we confirmed that our proposed method outperforms the other methods.

Chapter 5

Squeezed Semantic Segmentation for Fire Images

In this chapter, we propose a lightweight version of the semantic fire image segmentation model proposed in the previous chapter. We also introduce various compression methods and explain how to make the model lighter.

5.1 Related work

In 2016, Iandola *et al.* [21] proposed a version of SqueezeNet with similar performance but fewer parameters. The smaller the model, the easier it is to adapt to non-heavy devices such as mobile or embedded devices. The researchers replaced the 3×3 filter with a 1×1 filter, as it reduced the parameters by nine times. They also proposed Fire modules that use two different types of filters in the same layer. They compressed the classification model size by 510 times, and the performance is close to that of the original network.

MobileNet V1 [18] and V2 [32] were proposed in 2017 and 2018, respectively. The first version proposed an efficient model for mobile and embedded vision applications in image classification and object detection tasks. To build a lightweight network, it replaced convolutions with depthwise separable convolutions. The second version proposed an inverted residual module with a linear bottleneck. This module takes a low-dimensional compressed represen-

tation as input, expands it first to a higher dimension, passes it through a depthwise convolution filter, and then projects it back to a low-dimensional representation via linear convolution. It provides a highly efficient mobile-oriented model that can be used as a base for many visual recognition tasks, such as classification and segmentation.

5.2 Squeezed Fire Binary Segmentation Networks(SFBSNet)

In this section, we explain the details of our proposed model and its implementation, and how to reduce the parameters to apply it to embedded devices. We reduced the depth of the feature map and replaced the regular convolution with depthwise convolution to reduce the training parameters of the network proposed in the previous chapter. A detailed description of how to reduce the parameters is in the following section.

5.2.1 SFBSNet architecture

Feature map visualization is a powerful tool for understanding neural networks and examining how they work properly. Before performing feature map visualization following the method in [62], FusionNet, Base_model proposed in the previous chapter, and our network are trained sufficiently from the scratch for Corsican datasets. Each row of **Figure 5.1** shows the mean of feature map activation values without zeros for the layer at the end of each downscaling block. The number of mean values close to zero is high at the end layer of each downscaling block in FusionNet, and the ratio of the second downscaling block in particular is too high. In Base_model, the ratio is much lower than FusionNet at the second layer, but it is still high at third layer. It can be interpreted that Base_model has unnecessary parameters. The third row of **Figure 5.1** shows that our network has a lower ratio of values close to zero than Base_model and FusionNet.

Figure 5.2 shows the feature map visualizations from FusionNet (top row), Base_model (middle row), and our model (bottom row) proposed in this chapter once training is complete. Our model shows all the feature maps, and

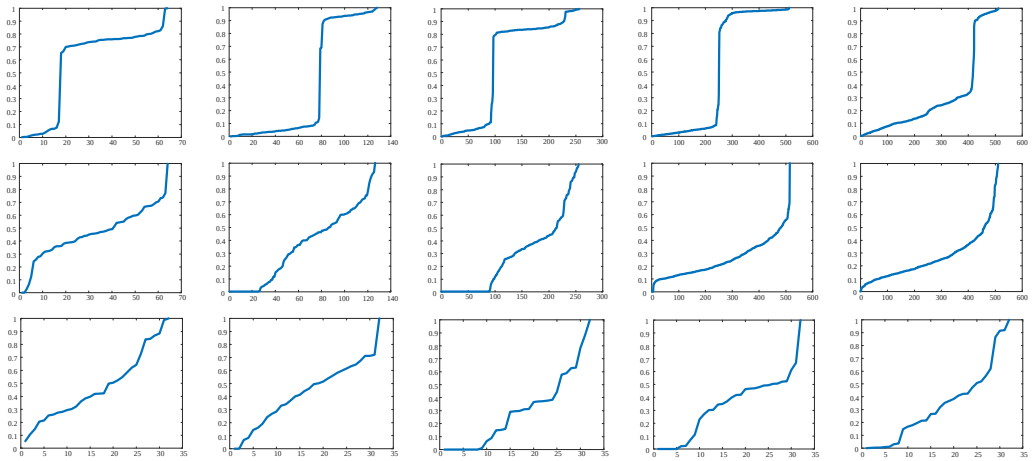

Figure 5.1: Sorted mean values of feature map activation values without zeros in the last layer of each downscaling block: FusionNet (1st row), Base_model (2nd row), ours (3rd row).

we selected only some of the feature maps in FusionNet and Base_model as there are so many of them. As mentioned earlier, FusionNet has a lot of zero values in its feature maps (all black squares), and the shapes of the feature maps are repetitive and they lack diversity. Base_model is much more diverse than FusionNet's feature maps, and the number of black square featuremaps is also reduced. Based on this visualization method, we can confirm that reducing unnecessary filters not only reduces parameters but also contributes to performance improvements in this task.

It also reduces parameters by replacing regular convolution with depthwise separable convolution. A depthwise separable convolution consists of a depthwise convolution, performed independently over each channel, and a pointwise convolution, projecting the output channel of the depthwise convolution onto a new channel space [8]. In **Figure 5.3**, the left operation is a depthwise convolution and the following operation is a pointwise convolution. The proposed network using depthwise separable convolution has fewer parameters and a much lighter model size than the model using regular convolution and Base_model, as shown in **Table 5.1**.

The proposed network (SFBSNet) has an encoding phase consisting of

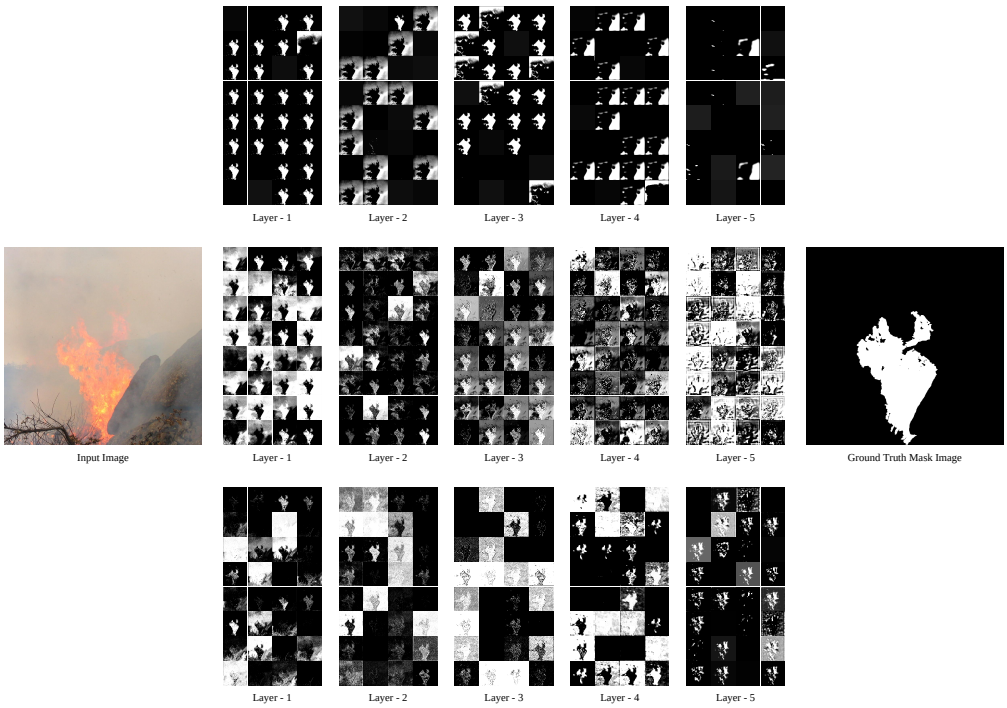

Figure 5.2: Visualization of features at the last layer of each downscaling block: FusionNet(top row), Base_model (middle row), and Ours(bottom row).

four downscaling blocks and a bridge block, and a decoding phase consisting of four upscaling blocks. Our model reduces the size of the feature map using a convolution stride2 as Base_model. The filter sizes of all the convolutions and deconvolutions used in this network are unified to 3×3 except for the 1×1 convolution block. Batch normalization and the activation function are then performed through all the blocks. All the activation functions used in this network are unified to the ReLU.

In our network, the CRC block is basically used for the encoding and decoding phases, and the composition is as follows. The feature map passes through the depthwise separable convolution, then through the residual block consisting of three depthwise separable convolutions with the short skip connection, and finally through the depthwise separable convolution. The depth of the feature map in all CRC blocks is equal to 64.

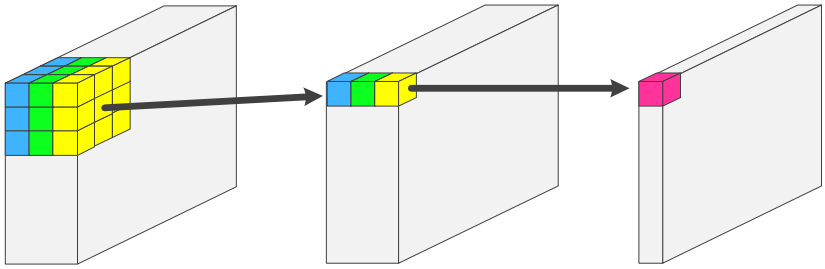

Figure 5.3: Description of depthwise separable convolution.

Networks	Compression method	Model size	# of parameters
Base_model	-	712 MB	72.63 m
Proposed_regular	1×1 Conv	17 MB	1.69 m
Proposed_separable	Depth Sep Conv	2.9 MB	0.27 m

Table 5.1: Comparing SFBSNet to other networks by approaching model compression method.

The encoding phase is composed as follows. First, the input image passes through the entrance convolution block and CRC block. Next, the output of the CRC block passes through the 1×1 convolution, and the depth of the feature map changes from 64 to 32. After this, the size of the feature map is reduced after passing through the stride-2 convolution. This process is repeated until the size of the feature map is 40×40. The final stage of the encoding phase, Bridge, is as follows. The feature map of the downscaling block of the last stage of the encoding phase passes through the CRC block and 1×1 convolution in sequence, and a feature map with a depth of 32 is output.

The decoding phase is composed as follows. The feature map, which is the result of 1×1 convolution, passes through the deconvolution block, and the deconvolution result is concatenated with the long skip connection with the previous feature map of the encoding phase. Next, this concatenated feature map passes the CRC block again and, as before, passes through the 1×1 convolution; the depth of the feature map then changes from 64 to 32. This process is repeated until the size of the feature map is 640×640, the same

as the original input shape. After this, the feature map that passed through the 1×1 convolution and size of 640×640 passes the exit convolution, and its depth is 32 to 1.

The details of our proposed method are shown in **Figure 5.4** and **Table 5.2**. In **Figure 5.4**, the green block is the 3×3 depthwise separable convolution, the purple block is the residual block, the yellow block is the 1×1 convolution, the blue block is the 3×3 depthwise separable convolution stride-2, and the red block is the deconvolution.

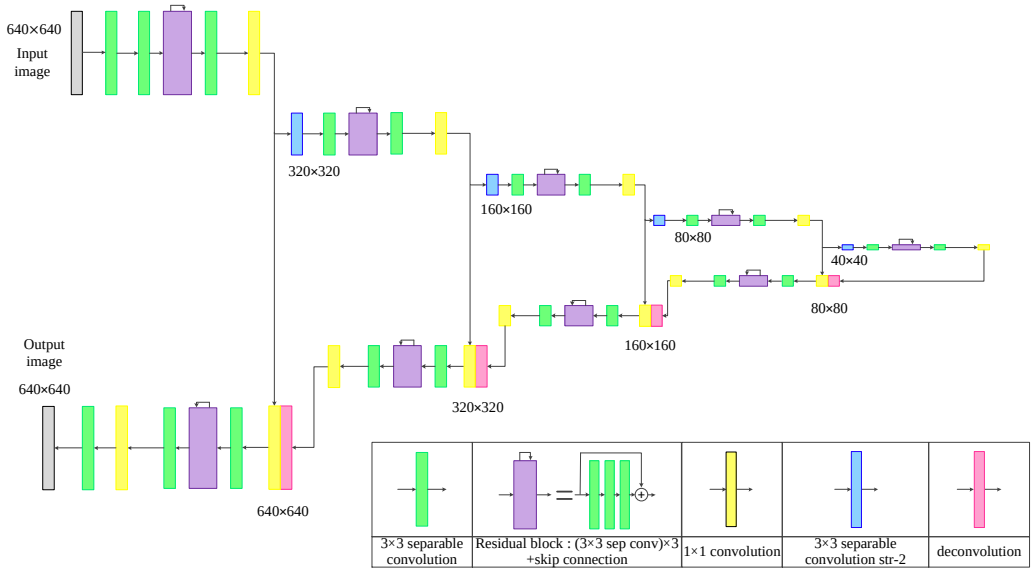

Figure 5.4: SFBSNet architecture.

5.2.2 Implementation details

We implement our network in Python and TensorFlow and train it with one GPU. The image is resized to 640×640 for training, and we use an Adam optimizer and mini-batch size of 2. The learning rate starts from 0.01 and is divided by ten specific iterations, and the models are trained until loss can be sufficiently converged. We train the network from scratch with pairs of

Block type	Ingredients	Size of feature maps
inputs		$640 \times 640 \times 3$
entrance conv	convolution	$640 \times 640 \times 64$
Downscaling-1	CRC block	$640 \times 640 \times 64$
	Conv1 block	$640 \times 640 \times 32$
	Conv str-2	$320 \times 320 \times 32$
Downscaling-2	CRC block	$320 \times 320 \times 64$
	Conv1 block	$320 \times 320 \times 32$
	Conv str-2	$160 \times 160 \times 32$
Downscaling-3	CRC block	$160 \times 160 \times 64$
	Conv1 block	$160 \times 160 \times 32$
	Conv str-2	$80 \times 80 \times 32$
Downscaling-4	CRC block	$80 \times 80 \times 64$
	Conv1 block	$80 \times 80 \times 32$
	Conv str-2	$40 \times 40 \times 32$
bridge	CRC block	$40 \times 40 \times 64$
	Conv1 block	$40 \times 40 \times 32$
Upscaling-4	DeConv, concate	$80 \times 80 \times 64$
	CRC block	$80 \times 80 \times 64$
	Conv1 block	$80 \times 80 \times 32$
Upscaling-3	DeConv, concate	$160 \times 160 \times 64$
	CRC block	$160 \times 160 \times 64$
	Conv1 block	$160 \times 160 \times 32$
Upscaling-2	DeConv, concate	$320 \times 320 \times 64$
	CRC block	$320 \times 320 \times 64$
	Conv1	$320 \times 320 \times 32$
Upscaling-1	DeConv, concate	$640 \times 640 \times 64$
	CRC block	$640 \times 640 \times 64$
	Conv1	$640 \times 640 \times 32$
Output	Conv	$640 \times 640 \times 1$

Table 5.2: Summary of proposed architecture.

Method (input image size)	Model size	# of par	FPS	FPS(TX2)
Deep Smoke [61] (256×256)	165 MB	43.2 m	27	1
Proposed_sep (256×256)	**2.2 MB**	**0.20 m**	**40**	**5**
FusionNet [41] (640×640)	791 MB	69.1 m	10	0.34
FCN+FC-ResNet [9] (512×512)	302 MB	26.3 m	12	0.73
YC_bC_r Color Feature [54] (None)	-	-	**24**	-
Proposed_regular (640×640)	17 MB	1.69 m	14	0.67
Proposed_sep (640×640)	**2.9 MB**	**0.27 m**	14	**0.80**

Table 5.3: Model specifications.

images each with the image and its label, compare the output with manual segmentation, and use the mean-square-error loss function to back-propagate to adjust the weights of the network.

5.3 Experiments

To verify that the network is also applicable to embedded devices, we use the embedded device NVIDIA Jetson TX2 board. It is built around an NVIDIA Pascal-family GPU and loaded with 8 GB of memory. The results illustrate that the running of our network on the Jetson TX2 embedded system is near real-time frame rates. **Figure 5.5** shows the NVIDIA Jetson TX2 kit and an example of the results.

Table 5.3 shows a comparison of model specifications between our methods and the others. As the Deep Smoke method takes an input size of 256×256, our model is also reduced to the same size for equivalent comparison. Our model shows overwhelmingly fewer parameters, a smaller model size, and higher frames per second(FPS) than the other methods.

In this section, we verified the proposed architecture through extensive experiments. Our experiments were performed on the FiSmo dataset and the Corsican Fire Database under the same conditions as in the previous chapter. As previously, 80% of the FiSmo dataset and Corsican Fire Database were used as training data. We compared our networks against the FusionNet [41], FCN+FC-ResNet [9], YC_bC_r Color Feature [54], and Deep Smoke

models [61] for the FiSmo dataset and Corsican Fire Database. Here, it was confirmed that the Deep Smoke method resulted in a generally smooth prediction form, and the YC_bC_r Color Feature method resulted in generally fragmented results as shown in **Figure 5.6** and **Figure 5.7**, as mentioned in the previous chapter. In addition, to further verify the validity of our model, we conducted additional experiments on the Still dataset, which was different from the training data.

As in the previous chapter, we used the IoU value as an evaluation metric to verify the similarity between the prediction results and the corresponding ground-truth images. As **Table 5.4** shows, our model also has a higher IoU value in the FiSmo dataset and Corsican Fire Database than the other models.

Figure 5.5: Experimental example in real life. Fire situation (left), Jetson TX2 with camera (middle), and segmentation results (right).

Method	FiSmo	Corsican
FusionNet [41]	0.63653529	0.77014998
FCN+FC-ResNet [9]	0.66398529	0.85016260
YC_bC_r Color Feature [54]	0.47071706	0.42878885
Deep Smoke [61]	0.59600632	0.76479557
SFBSNet(proposed_regular)	0.73005690	0.89348606
SFBSNet(proposed_separable)	**0.78375529**	**0.90698019**

Table 5.4: Comparison of IoU metric results with other methods – FiSmo dataset and Corsican Fire Database.

5.3.1 Ablation study

This subsection presents the ablation study on FiSmo datasets. We show the effectiveness of the addition and concatenation in our networks in **Table 5.5**. We first included the addition operation to the input and output features of the residual block in each downscaling and upscaling block. Our network with addition in **Table 5.5** shows that the IoU improves by 0.03 without addition. We then checked whether the concatenation in long skip connections is effective. Earlier, when we replaced the concatenation operation with the addition operation, such as that implemented in FusionNet, the IoU was lower than that of the proposed model. Lastly, when we removed the concatenation, the IoU was still lower than that of the proposed model.

Method	IoU	Method	IoU
w/ addition (middle skip)	0.75036249	con2add (long skip)	0.75192128
w/o concatenate	0.73622126	ours_separable	**0.78375529**

Table 5.5: Ablation experiment for FiSmo dataset

5.3.2 Experiments on FiSmo dataset

In this case, training was conducted effectively by setting a learning rate schedule of 10^{-2} for iterations [1, 500], 10^{-3} for iterations [501, 1500], 10^{-4} for iterations [1501, 2500], 10^{-5} for iterations [2501, 5000], 5×10^{-6} for iterations [5001, 7500], and 10^{-6} for higher iterations. The results obtained through this method are as follows.

The first, second, fourth, and sixth rows of **Figure 5.6** are the results for the fire096.png, fire097.png, fire115.png, and fire118.png images, respectively. FusionNet and FCN+FC-ResNet lost detailed information or had misclassified information, as indicated by the red circles. However, the proposed method best captured the real fire area and minimized misclassification.

The third row of **Figure 5.6** is the result for the fire112.png image. For YC_bC_r, some details were lost, and FCN+FC-ResNet had too many misclassification results in fire segmentation. However, the proposed architecture yields better classification results.

The fifth row of **Figure 5.6** is the result for the fire117.png image. For FusionNet and FCN+FC-ResNet, as indicated by the yellow arrow, the fireman's fire suit was misclassified as fire because of its color. However, the proposed method achieved the most similar images to the ground-truth images overall.

The average values of the IoU obtained using the FiSmo dataset are shown in **Table 5.4**. Through this, it was found that the proposed method, at its most effective, outperformed the other methods.

Figure 5.6: FiSmo image results. The order of the figure is 96, 97, 112, 115, 117, 118.

5.3.3 Experiments on Corsican Fire Database

In this case, training on the Corsican Fire Database was conducted effectively by setting a learning rate schedule as 10^{-2} for iterations [1, 1000], 10^{-3} for iterations [1001, 3000], 10^{-4} for iterations [3001, 5000], 10^{-5} for iterations [5001, 15000], 5×10^{-6} for iterations [15,001, 20,000], and 10^{-6} for higher iterations. The results obtained through this method are as follows.

The first row of **Figure 5.7** is the result for the 408_rgb.png image. FusionNet incorrectly classified the bright areas in the forest as fire, and $\mathrm{YC}_b\mathrm{C}_r$ missed the inside of the area of fire.

The second row of **Figure 5.7** is the result for the 414_rgb.png image. As we know from the result images, as indicated by the red circles, all the methods used in the study recognized the outfit of a firefighter as the correct flame result. However, the proposed method best captured the real fire area and minimized misclassification.

The remaining rows of **Figure 5.7** are the results for the 422_rgb.png, 423_rgb.png, 434_rgb.png, and 493_rgb.png images, respectively. From the FusionNet results it can be observed that false classification results were generated near the fire, and for FCN+FC-ResNet, there is a hole inside the fire prediction area. However, the proposed architecture yields better classification results.

The average values of the IoU obtained using the Corsican Fire Database are shown in **Table 5.4**. Through this, it was found that the proposed method, at its most effective, outperformed the other methods.

5.3.4 Additional experiments on Still dataset

The experiment was also applied to the Still dataset used in fire detection, which is a completely different type of data from the FiSmo dataset and Corsican Fire Database. Before applying the model to the Still dataset, it was trained using both of the previous datasets, and the default settings were the same as before. As the FiSmo dataset and Corsican Fire Database include ground-truth images, IoU metrics can be used to measure accuracy. However, for the Still dataset the metric cannot be used because there are no ground-truth images. Therefore, to evaluate the performance, it is necessary

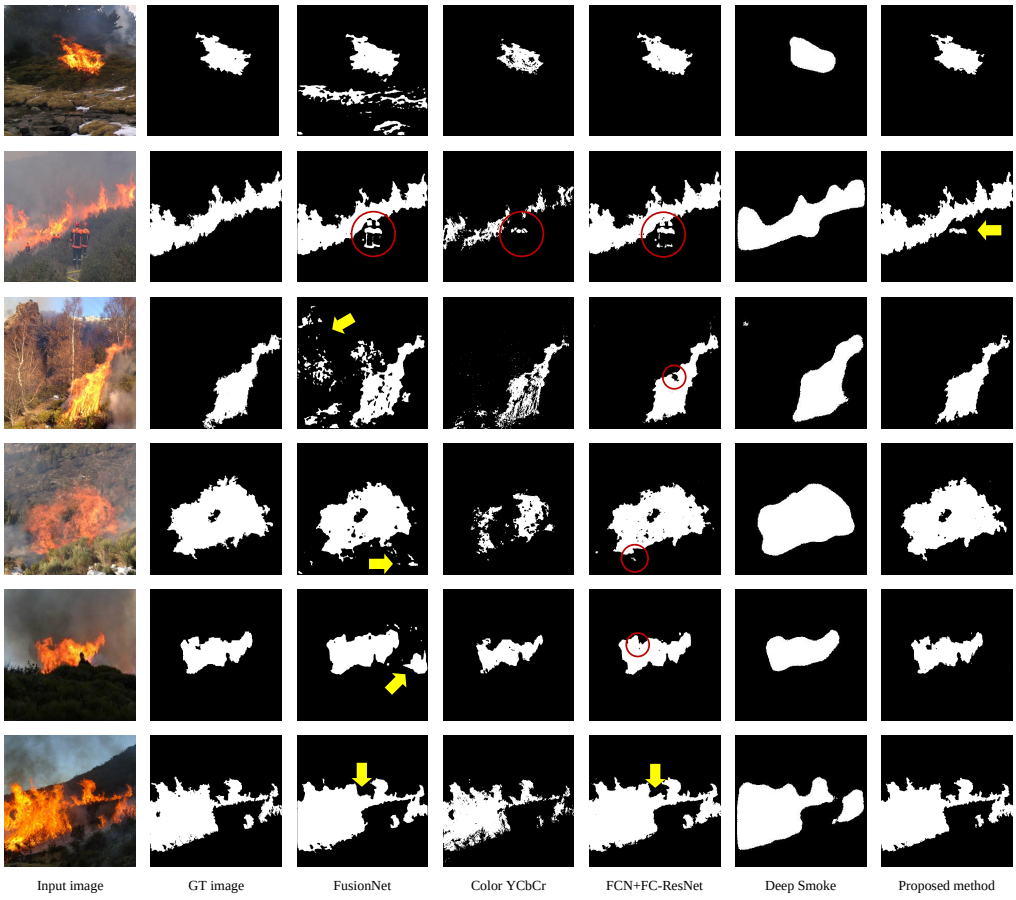

Figure 5.7: Corsican Fire Database image results. The order of the figure is 408, 414, 422, 423, 434, 493.

to check the prediction results of images in various environments. **Figure 5.8** shows the prediction results for the Still dataset. Although the environment is different from the training data, our model captures the area of fire properly, except for the very small area indicated by the red circle.

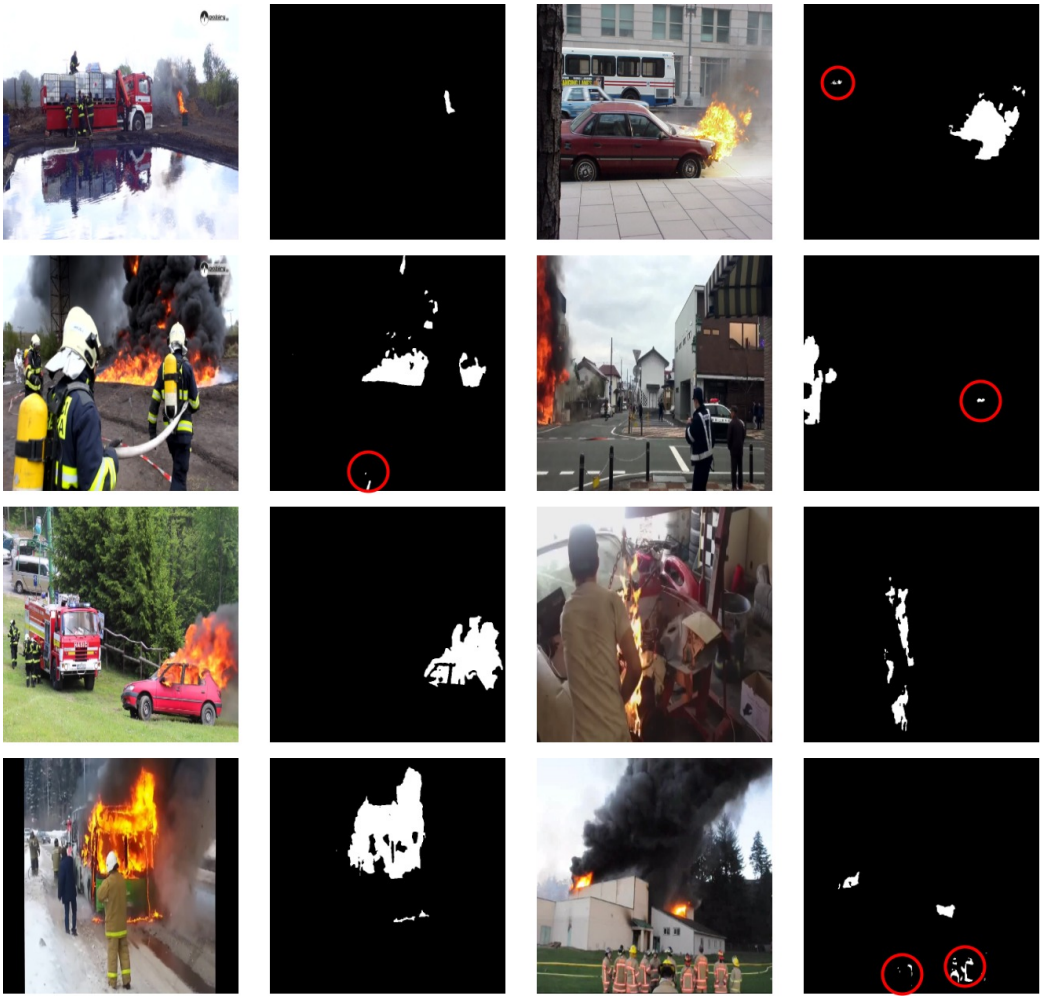

Figure 5.8: Results for the Still dataset.

5.4 Conclusion

In this chapter, we propose a powerful yet lightweight network for the binary segmentation task for fire image data. The proposed model reduces features at each stage compared to the existing binary segmentation methods, and uses depthwise separable convolution instead of regular convolution to further reduce variables as well as improve performance. Extensive experimental results show that the proposed method is able to execute on an embedded

device with good results. Based on these results, we can see that our model not only works in an environment with less computing power but also shows good performance. Therefore, we can expect the proposed model to be used as a reliable aid for predicting fire areas when a fire accident occurs.

Chapter 6

Conclusion and Future Works

In this thesis, we introduced various kinds of deep-learning models for image classification, object detection, and segmentation, and proposed new networks for fire detection and segmentation. First, we proposed a new fire detection model by combining Res2Net and ResNet blocks, which showed higher performance than the existing methods. In addition, to secure the validity of the proposed fire detection model, the area influencing the classification criteria of the model was confirmed using Grad-CAM. Moreover, the insufficient fire detection performance of the localization model of the fire area was improved by supplementing it with the proposed network. Second, we proposed a fire image segmentation network and its compression model, both of which performed better than the existing methods. In particular, the compression model is much lighter than the existing method, so it is fast and can be easily applied to embedded devices. We propose to continue with this study and in the future will propose a model for fire detection, localization, and segmentation in one network.

Bibliography

[1] Süleyman Aslan, Uğur Güdükbay, B Uğur Töreyin, and A Enis Çetin (2019). Deep convolutional generative adversarial networks based flame detection in video. *arXiv preprint arXiv:1902.01824*.

[2] Robert Bogue (2013, 03). Sensors for fire detection. *Sensor Review 33*.

[3] Yiheng Cai, Yajun Guo, Yuanyuan Li, Hui Li, and Jiaqi Liu (2019, 10). Fire Detection Method Based on Improved Deep Convolution Neural Network. pp. 466–470.

[4] Mirela Cazzolato, Letricia Avalhais, Daniel Chino, Jonathan Ramos, Jessica Souza, Jose Rodrigues Jr, and Agma Taina (2017, 10). FiSmo: A Compilation of Datasets from Emergency Situations for Fire and Smoke Analysis.

[5] Turgay Celik (2010). Fast and Efficient Method for Fire Detection Using Image Processing. *ETRI Journal 32*(6), 881–890.

[6] Shin-Juh Chen, David Hovde, Kristen Peterson, and André Marshall (2007, 11). Fire detection using smoke and gas sensors. *Fire Safety Journal 42*, 507–515.

[7] Daniel Chino, Letricia Avalhais, Jose Rodrigues Jr, and Agma Traina (2015, 08). BoWFire: Detection of Fire in Still Images by Integrating Pixel Color and Texture Analysis. pp. 95–102.

[8] François Chollet (2017, July). Xception: Deep Learning with Depthwise Separable Convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807.

BIBLIOGRAPHY

[9] Michal Drozdzal, Gabriel Chartrand, Eugene Vorontsov, Mahsa Shakeri, Lisa Di Jorio, An Tang, Adriana Romero, Yoshua Bengio, Chris Pal, and Samuel Kadoury (2018). Learning normalized inputs for iterative estimation in medical image segmentation. *Medical Image Analysis 44*, 1 – 13.

[10] Michal Drozdzal, Eugene Vorontsov, Gabriel Chartrand, Samuel Kadoury, and Chris Pal (2016). The Importance of Skip Connections in Biomedical Image Segmentation. *CoRR abs/1608.04117*.

[11] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman (2009). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision 88*, 303–338.

[12] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C. Berg (2017). DSSD : Deconvolutional Single Shot Detector. *CoRR abs/1701.06659*.

[13] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xinyu Zhang, Ming-Hsuan Yang, and Philip H. S. Torr (2019). Res2Net: A New Multi-scale Backbone Architecture. *CoRR abs/1904.01169*.

[14] Ross Girshick (2015). Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448.

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587.

[16] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik (2014). Simultaneous Detection and Segmentation. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014*, Cham, pp. 297–312. Springer International Publishing.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016, 10). Identity Mappings in Deep Residual Networks. Volume 9908, pp. 630–645.

BIBLIOGRAPHY

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. cite arxiv:1704.04861.

[19] Jie Hu, Li Shen, and Gang Sun (2017). Squeeze-and-Excitation Networks. *CoRR abs/1709.01507*.

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger (2017). Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2261–2269.

[21] Forrest Iandola, Matthew Moskewicz, Khalid Ashraf, Song Han, William Dally, and Kurt Keutzer (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *CoRR abs/1602.07360*.

[22] Arpit Jadon, Mohd. Omama, Akshay Varshney, Mohammad Samar Ansari, and Rishabh Sharma (2019). FireNet: A Specialized Lightweight Fire & Smoke Detection Model for Real-Time IoT Applications. *CoRR abs/1905.11922*.

[23] Thomas Kaiser (2000, Oct). Fire detection with temperature sensor arrays. In *Proceedings IEEE 34th Annual 2000 International Carnahan Conference on Security Technology (Cat. No.00CH37083)*, pp. 262–268.

[24] Dong-Keun Kim (2009, 01). Flame Detection using Region Expansions and On-line Variances in Infrared image. *Journal of Korea Multimedia Society 12*.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (2012, 01). ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems 25*.

[26] Ye Li, Lele Xu, Jun Rao, Lili Guo, Zhen Yan, and Shan Jin (2019). A Y-Net deep learning method for road segmentation using high-resolution visible remote sensing images. *Remote Sensing Letters 10*(4), 381–390.

[27] Gaohua Lin, Yongming Zhang, Gao Xu, and Qixing Zhang (2019). Smoke detection on video sequences using 3D convolutional neural networks. *Fire Technology 55*(5), 1827–1847.

[28] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár (2017). Focal Loss for Dense Object Detection. *CoRR abs/1708.02002*.

[29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick (2014). Microsoft COCO: Common Objects in Context. *CoRR abs/1405.0312*.

[30] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg (2015). SSD: Single Shot MultiBox Detector. *CoRR abs/1512.02325*.

[31] Jonathan Long, Evan Shelhamer, and Trevor Darrell (2015, June). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440.

[32] Menglong Zhu Andrey Zhmoginov Liang-Chieh Chen Mark Sandler, Andrew Howard (2018, June). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[33] Jozef Mlích, Karel Koplík, Michal Hradiš, and Pavel Zemčík (2020). Fire Segmentation in Still Images. In J. Blanc-Talon, P. Delmas, W. Philips, D. Popescu, and P. Scheunders (Eds.), *Advanced Concepts for Intelligent Vision Systems*, Cham, pp. 27–37. Springer International Publishing.

[34] Khan Muhammad, Jamil Ahmad, Irfan Mehmood, Seungmin Rho, and Sung Baik (2018, 03). Convolutional Neural Networks based Fire Detection in Surveillance Videos. *IEEE Access PP*, 1–1.

[35] Khan Muhammad, Salman Khan, Mohamed Elhoseny, Syed Ahmed, and Sung Baik (2019a, 02). Efficient Fire Detection for Uncertain Surveillance Environment. *IEEE Transactions on Industrial Informatics PP*, 1–1.

[36] Khan Muhammad, Salman Khan, Mohamed Elhoseny, Syed Hassan Ahmed, and Sung Wook Baik (2019b). Efficient fire detection for uncertain surveillance environment. *IEEE Transactions on Industrial Informatics 15*(5), 3113–3122.

[37] Yuval Nirkin, Iacopo Masi, Anh Tuan Tran, Tal Hassner, and Gérard G. Medioni (2017). On Face Segmentation, Face Swapping, and Face Perception. *CoRR abs/1704.06729*.

[38] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han (2015). Learning Deconvolution Network for Semantic Segmentation. In *Computer Vision (ICCV), 2015 IEEE International Conference on*.

[39] C Emmy Premal and SS Vinsley (2014). Image processing based forest fire detection using YCbCr colour model. In *2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014]*, pp. 1229–1237. IEEE.

[40] Xuanbing Qiu, Tingyu Xi, Dongyuan Sun, Enhua Zhang, Chuanliang Li, Ying Peng, Jilin Wei, and Gao Wang (2018, Sep). Fire Detection Algorithm Combined with Image Processing and Flame Emission Spectroscopy. *Fire Technology 54*(5), 1249–1263.

[41] Tran Minh Quan, David Hildebrand, and Won-Ki Jeong (2016). FusionNet: A deep fully residual convolutional neural network for image segmentation in connectomics.

[42] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi (2015). You Only Look Once: Unified, Real-Time Object Detection. *CoRR abs/1506.02640*.

[43] Joseph Redmon and Ali Farhadi (2018). YOLOv3: An Incremental Improvement. *CoRR abs/1804.02767*.

[44] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *CoRR abs/1506.01497*.

[45] Olaf Ronneberger, Philipp Fischer, and Thomas Brox (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. Volume abs/1505.04597, pp. 234–241.

[46] Faisal Saeed, Anand Paul, P. Karthigaikumar, and Anand Nayyar (2019, 04). Convolutional neural network based early fire detection. *Multimedia Tools and Applications 79*, 9083–9099.

[47] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra (2016). Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. *CoRR abs/1610.02391*.

[48] Jivitesh Sharma, Ole-Christoffer Granmo, Morten Goodwin, and Jahn Thomas Fidje (2017). Deep Convolutional Neural Networks for Fire Detection in Images. In G. Boracchi, L. Iliadis, C. Jayne, and A. Likas (Eds.), *Engineering Applications of Neural Networks*, Cham, pp. 183–193. Springer International Publishing.

[49] Karen Simonyan and Andrew Zisserman (2014, 09). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv 1409.1556*.

[50] S Sudhakar, V Vijayakumar, C Sathiya Kumar, V Priya, Logesh Ravi, and V Subramaniyaswamy (2020). Unmanned Aerial Vehicle (UAV) based Forest Fire Detection and monitoring for reducing false alarms in forest-fires. *Computer Communications 149*, 1–16.

[51] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9.

[52] Mingxing Tan and Quoc V. Le (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *CoRR abs/1905.11946*.

[53] Tom Toulouse, Lucile Rossi, Antoine A Campana, Turgay A Celik, and Moulay A Akhloufi (2017). Computer vision for wildfire research: An evolving image dataset for processing and analysis. *Fire Safety Journal 92*, 188–194.

[54] Viktor Tuba, Romana Capor-Hrosik, and Eva Tuba (2017). Forest Fires Detection in Digital Images Based on Color Features. *International Journal of Environmental Science 2*, 66–70.

[55] Behçet Töreyin, Yigithan Dedeoglu, U. Gudukbay, and A. Cetin (2006, 01). Computer vision based method for real-time fire and flame detection. *Pattern Recognition Letters 27*, 49–58.

[56] Andreas Veit, Michael Wilber, and Serge Belongie (2016, 05). Residual Networks Behave Like Ensembles of Relatively Shallow Networks. *Advances in Neural Information Processing Systems*.

[57] K. Villela, K. Breiner, Claudia Nass, M. Mendonça, and V. Vieira (2014, 01). A smart and reliable crowdsourcing solution for emergency and crisis management. *IDIMT 2014: Networking Societies - Cooperation and Conflict, 22nd Interdisciplinary Information Management Talks*, 213–220.

[58] Michael Wurm, Thomas Stark, Xiao Xiang Zhu, Matthias Weigand, and Hannes Taubenböck (2019). Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing 150*, 59 – 69.

[59] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He (2016). Aggregated Residual Transformations for Deep Neural Networks. *CoRR abs/1611.05431*.

[60] Hisao Yamagishi and Jun'ichi Yamaguchi (1999). Fire flame detection algorithm using a color camera. *MHS'99. Proceedings of 1999 International Symposium on Micromechatronics and Human Science (Cat. No.99TH8478)*, 255–260.

[61] Feiniu Yuan, Lin Zhang, Xue Xia, Boyang Wan, Qinghua Huang, and Xuelong Li (2019). Deep smoke segmentation. *Neurocomputing 357*, 248 – 260.

[62] Matthew D. Zeiler and Rob Fergus (2014). Visualizing and Understanding Convolutional Networks. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014*, Cham, pp. 818–833. Springer International Publishing.

[63] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba (2015). Learning Deep Features for Discriminative Localization. *CoRR abs/1512.04150*.

[64] Turgay Çelik and Hasan Demirel (2009). Fire detection in video sequences using a generic color model. *Fire Safety Journal 44*(2), 147 – 158.

국문초록

최근 딥 러닝은 다양한 분야에서 가장 중요하고 강력한 주제이다. 딥러닝은 이미지 분류에서 뛰어난 성능을 보였으며, 이후 컴퓨터 비전의 이지미에서 객체 감지 및 시맨틱 분할에도 적용되었다. 본 논문에서는 뛰어난 성능을 가진 딥 러닝을 사용하여 화재 이미지 감지 및 분할 작업에 적합한 네트워크를 제안한다. 또한 딥러닝 압축 기법을 사용하여 화재 이미지 분할 딥러닝 모델에 적용하여 소규모 네트워크를 제안하였고 이를 임베디드 장치에 적용했다. 여러가지 광범위한 실험을 통해 화재감지와 화재 이미지 분할에서 기존의 기법보다 좋다는 점을 보였다.

주요어휘: 이미지 시맨틱 분할, 객체 감지, 딥 러닝, 화재 이미지, 딥러닝 압축
학번: 2014-31032